TECHNISCHE UNIVERSITÄT MÜNCHEN

Lehrstuhl für Flugsystemdynamik

Trajectory Planning in Time-Varying Adverse Weather for Fixed-Wing Aircraft applying a Model Predictive Approach

Dipl.-Ing.(FH) Federico Mothes

Vollständiger Abdruck der von der Fakultät für Luftfahrt, Raumfahrt und Geodäsie der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:		Prof. Dr. rer. nat. Ulrich Walter
Prüfer der Dissertation:	1.	Prof. Dr.-Ing. Florian Holzapfel
	2.	Prof. Dr.-Ing. Alexander Knoll, Hochschule München
	3.	Prof. Dr.-Ing. Axel Schulte, Universität der Bundeswehr München

Die Dissertation wurde am 19.05.2020 bei der Technischen Universität München eingereicht und durch die Fakultät für Luftfahrt, Raumfahrt und Geodäsie am 08.02.2021 angenommen.

Abstract

The avoidance of adverse weather is an important safety-relevant task in aviation. Automated avoidance can help to improve safety and profitability in manned and unmanned aviation. For this purpose, the Model Predictive Trajectory Planner (MPTP) is introduced, which is aimed at solving single-source-single-target motion planning problems amid moving obstacles. The functional principle is explained and tested in scenarios with time-varying polygonal obstacles, based on external thunderstorm nowcast. The presented resolution-complete combinatorial planner uses deterministic state sampling to continuously provide globally near-time-optimal trajectories for the predicted case. Inherent uncertainty in the prediction of dynamic environments is implicitly taken into account by a closed feedback loop and explicitly by bounded margins. The planner is able to anticipate and avoid future obstacles, while flying inside a restricted mission area. The computed trajectories are time-monotone and meet the nonholonomic turning-flight constraint of fixed-wing aircraft and therefore do not require postprocessing. Furthermore, the planner is capable of considering a time-varying goal and automatically plan holding patterns. Besides of the application for weather avoidance, it is suitable for all kinds of nonholonomic planning problems in the presence of nonlinear moving obstacles, whose motion cannot be described analytically. Due to its deterministic, reliable and robust properties, the MPTP provides optimum preconditions for a certification.

Zusammenfassung

Die Vermeidung schlechten Wetters ist eine wichtige, sicherheitsrelevante Aufgabe in der Luftfahrt. Eine automatische Wettervermeidung kann dazu beitragen, die Sicherheit und Kosteneffizienz, sowohl in der bemannten als auch in der unbemannten Luftfahrt, zu steigern. Zu diesem Zweck wird der Model Predictive Trajectory Planner (MPTP) vorgestellt, der darauf ausgelegt ist Flugbahnen für sog. single-source-single-target-Probleme mit bewegten Hindernissen zu planen. Es wird das Funktionsprinzip vorgestellt und in verschiedenen Szenarien, mit zeitvarianten Gewittern getestet, deren Daten von einer externen Gewittervorhersage stammen. Der auflösungsvollständige, kombinatorische Planer verwendet eine deterministische Zustandsabtastung, um kontinuierlich global nah-optimale Trajektorien für den erwarteten Fall zu berechnen. Die immanente Unsicherheit bei der Vorhersage dynamischer Umgebungen, wird implizit durch eine geschlossene Rückkopplung und explizit durch diskrete Sicherheitsmargen berücksichtigt. Bewegte Hindernisse können vorausschauend vermieden werden, ohne dabei ein vorgeschriebenes Missionsgebiet zu verlassen. Geplante Trajektorien sind zeitmonoton und berücksichtigen die nichtholonomen, dynamischen Beschränkungen eines Flächenflugzeugs. Daher benötigen diese auch keinerlei Nachbearbeitung. Außerdem ist der Algorithmus in der Lage, ein zeitveränderliches Ziel vorauschauend anzusteuern und automatisch Warteschleifen zu planen. Neben der Wettervermeidung, kann die vorgestellte Planungsmethode auch auf andere Arten von nichtholonomen Planungsproblemen anwendet werden, bei denen die nichtlineare Dynamik der Hindernisse nicht analytisch beschrieben werden kann. Dank seiner deterministischen, robusten und zuverlässigen Eigenschaften bringt der MPTP optimale Voraussetzungen für eine Zulassung mit sich.

Danksagung

Zu Beginn möchte ich mich ganz herzlich bei meinen beiden Betreuern Florian Holzapfel und Alexander Knoll (alphabetische Reihenfolge der Nachnamen) bedanken. Sie haben mir die optimalen Rahmenbedingungen geboten, um diese Arbeit zu verfassen. Insbesondere fachliche Unterstützung und kreativer Freiraum waren dabei besonders hilfreich.

Ulrich Walter danke ich dafür, dass er so bereitwillig meinen Prüfungsvorsitz übernommen und für einen reibungslosen Ablauf gesorgt hat.

Rainer Nehrlich und Jürgen Maier gebührt Dank für Ihre Unterstützung bei unseren Forschungsprojekten und damit meiner Forschung und Finanzierung.

Bei meiner Freundin Katja Mitzscherling möchte ich mich besonders für ihre großartige und tatkräftige Unterstützung bezüglich meiner Promotion bedanken.

Bei meiner Freundin Monica Kleinoth-Gross bedanke ich mich für die exklusiven Termine bei Florian, die vielen wertvollen Ratschläge und Gespräche.

Andreas Klöckner danke ich dafür, dass er mich auf meine erste Konferenz begleitet und mir dort geholfen hat meinen Vortragsstil zu entwickeln.

Besonderer Dank gebührt meinen Freunden Reiko Müller, Ferdinand Settele, Alexander Weber (alphabetische Reihenfolge der Nachnamen) für Ihre großartige Betreuung und Unterstützung. Bessere Kollegen kann man sich einfach nicht wünschen!

Bei meinem Großvater Ricardo bedanke ich mich dafür, dass er mich darin bestätigt hat diesen Weg zu beschreiten und stets an mich geglaubt hat.

Meinen lieben Schwiegereltern Renate und Helmut möchte ich dafür danken, dass sie mich in allen Lebenslagen mit viel Rat und Tat unterstützt haben.

Meine Eltern Maria-Alicia und Gerardo haben immer für mich gesorgt, immer an mich geglaubt und mich immer unterstützt: die Grundlage für meinen Erfolg im Leben.

Meine geliebte Ehefrau Claudia und meine Kinder haben mir die notwendige Motivation, Inspiration und Ausdauer zur Erstellung dieser Dissertation verliehen.

Contents

List of Figures

List of Tables

Abbreviations, Symbols and Indices

Abbreviations

3-DoF	Three Degree of Freedom
6-DoF	Six Degree of Freedom
CAS	Computer Algebra System
CG	Center of Gravity
CPU	Central Processing Unit
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DLR	Deutsches Zentrum für Luft- und Raumfahrt
DWD	Deutscher Wetterdienst
EDH	Euclidean Distance Heuristic
EXST	Explored States
FAA	Federal Aviation Administration
FMM	Fast Marching Method
FSD	Institute of Flight System Dynamics
GC	Geometric Center
GRIB	General Regularly-distributed Information in Binary Form
HAPS	High Altitude Pseudo-Satellite
ISA	International Standard Atmosphere
MCS	Monte Carlo Simulation
MPC	Model Predictive Controller
MPTP	Model Predictive Trajectory Planner
MSFMM	Multistencils Second Order Fast Marching Method

MSL	Mean Sea Level
NATO	North Atlantic Treaty Organization
NGA	No Go Area
OUM	Ordered Upwind Method
PSPM	Partial Shortest Path Map
PSTM	Partial Shortest Trajectory Map
Rad-TRAM	Radar Tracking and Monitoring
RAM	Random-Access Memory
RK	Runge-Kutta
ROT	Rate One Turn
RRT	Rapidly-exploring Random Tree
SAA	Sense and Avoid
SPM	Shortest Path Map
SSPH	Shortest Static Path Heuristic
TRST	Trajectory States
TS	Taylor Series
TUM	Technische Universität München
WCET	Worst Case Execution Time
WGS84	World Geodetic System 1984
XML	Extensible Markup Language Format

Latin Symbols

$\boldsymbol{A}$	Set of Aircraft States
$\boldsymbol{ACD}$	Aircraft Covered Distance
$\boldsymbol{b}$	Branching Factor, in Context of Graph-Search
$\boldsymbol{b}$	Wingspan, in Context of Flight Dynamics
$\bar{\boldsymbol{c}}$	Mean Aerodynamic Chord
$\boldsymbol{C}$	Dimensionless Derivative, in Context of Flight Dynamics
$\boldsymbol{C}$	Set of Possible Configurations, in Context of Motion Planning

$\boldsymbol{CL}$	Closed List
$\boldsymbol{d}$	Distance
$\boldsymbol{D}$	Drag
$\boldsymbol{e}$	Edge of a Visibility Graph
$\boldsymbol{E}$	Set of Edges of a Visibility Graph
$\boldsymbol{ECA}$	Estimated Conflict Area
$\boldsymbol{ESA}$	Estimated Safe Area
$\boldsymbol{EX}$	Set of Estimated States
$\boldsymbol{F}$	Total Estimated Cost from Start to Goal
$\boldsymbol{g}$	Earth's Gravitation
$\boldsymbol{G}$	Cost to Go
$\boldsymbol{H}$	Heuristic Cost to the Goal
$\boldsymbol{I}$	Moment of Inertia
$\boldsymbol{ICO}$	Inevitable Collision Obstacle
$\boldsymbol{ICS}$	Inevitable Collision State
$\boldsymbol{L}$	Lift
$\boldsymbol{LD}$	Leg Distance
$\boldsymbol{m}$	Mass
$\boldsymbol{M}$	Matrix
$\mathbf{Ma}$	Mach Number
$\boldsymbol{MSD}$	Minimum Stabilization Distance
$\boldsymbol{n}$	Load Factor, in Context of Flight Dynamics
$\boldsymbol{np}$	Number of Neighbor Points
$\boldsymbol{O}$	Set of Obstacles
$\boldsymbol{OL}$	Open List
$\boldsymbol{p}$	Position, in Context of Motion Planning
$\boldsymbol{p}$	Roll Rate, around X-Axis of the Aircraft's Frame of Reference
$\boldsymbol{P}$	Engine Power, in Context of Flight Dynamics
$\boldsymbol{P}$	Probability, in Context of Probabilistic Margins
$\boldsymbol{q}$	Configuration of an Aircraft, in Context of Motion Planning

$\boldsymbol{q}$	Pitch Rate, around Y-Axis of the Aircraft's Frame of Reference
$\boldsymbol{\bar{q}}$	Dynamic Pressure
$\boldsymbol{r}$	Yaw Rate, around Z-Axis of the Aircraft's Frame of Reference
$\boldsymbol{R}$	Radius
$\boldsymbol{S}$	Sparsity, in Context of a Matrix
$\boldsymbol{S}$	Wing Area, in Context of Flight Dynamics
$\boldsymbol{t}$	Point in Time
$\boldsymbol{T}$	Period of Time
$\boldsymbol{TL}$	Trajectory Length
$\boldsymbol{u}$	Velocity, in the X-Axis of the Aircraft's Frame of Reference
$\boldsymbol{v}$	Velocity, in the Y-Axis of the Aircraft's Frame of Reference
$\boldsymbol{v}$	Vertex, in Context of a Visibility Graph
$\boldsymbol{V}$	Velocity, in Context of Flight Dynamics
$\boldsymbol{V}$	Set of Vertices, in Context of a Visibility Graph
$\boldsymbol{VG}$	Visibility Graph
$\boldsymbol{w}$	Velocity, in the Z-Axis of the Aircraft's Frame of Reference
$\boldsymbol{W}$	Wind Speed Vector, in Context of Flight Dynamics
$\boldsymbol{W}$	Workspace, in Context of Motion Planning
$\mathbf{x}$	State of an Aircraft
$\boldsymbol{X}$	Set of Possible States

Greek Symbols

$\boldsymbol{\alpha}$	Angle of Attack
$\boldsymbol{\beta}$	Sideslip Angle
$\boldsymbol{\gamma}$	Angle of Climb
$\boldsymbol{\epsilon}$	Error
$\boldsymbol{\zeta}$	Rudder
$\boldsymbol{\eta}$	Elevator
$\boldsymbol{\Theta}$	Pitch Angle

$\boldsymbol{\lambda}$	Longitude
$\boldsymbol{\mu}$	Bank Angle
$\boldsymbol{\xi}$	Aileron
$\boldsymbol{\tau}$	Roll Time Constant
$\boldsymbol{\varphi}$	Latitude
$\boldsymbol{\Phi}$	Bank Angle
$\boldsymbol{\chi}$	Course Angle
$\boldsymbol{\Psi}$	Azimuth Angle

Indices

$\boldsymbol{a}$	Adjacent
$\boldsymbol{A}$	Aerodynamic Frame of Reference
$\boldsymbol{ari}$	Approximate Roll In
$\boldsymbol{aro}$	Approximate Roll Out
$\boldsymbol{aux}$	Auxiliary
$\boldsymbol{B}$	Body Frame of Reference
$\boldsymbol{BE}$	From E- to B-Frame of Reference
$\boldsymbol{bt}$	Bitangent
$\boldsymbol{bta}$	Bitangent Edge Angle
$\boldsymbol{btc}$	Bitangent Candidate
$\boldsymbol{by}$	Fly-By
$\boldsymbol{c}$	Candidate
$\boldsymbol{cmd}$	Commanded
$\boldsymbol{cor}$	Corrected
$\boldsymbol{e}$	Earth
$\boldsymbol{E}$	Inertial Earth Fixed Frame of Reference
$\boldsymbol{EB}$	From B- to E-Frame of Reference
$\boldsymbol{fin}$	Final
$\boldsymbol{free}$	Free Region

$\boldsymbol{fut}$	Future
$\boldsymbol{g}$	Goal
$\boldsymbol{G}$	Ground Speed
$\boldsymbol{I}$	Indicated Airspeed
$\boldsymbol{in}$	Initial
$\boldsymbol{lat}$	Lateral
$\boldsymbol{lat_r}$	Relative Lateral
$\boldsymbol{lev}$	Level Flight
$\boldsymbol{max}$	Maximum
$\boldsymbol{min}$	Minimum
$\boldsymbol{N}$	Nowcast
$\boldsymbol{na}$	Next Edge Angle
$\boldsymbol{norm}$	Normalized
$\boldsymbol{np}$	Neighbor Points
$\boldsymbol{nr}$	No Roll
$\boldsymbol{obs}$	Obstacle Region
$\boldsymbol{ov}$	Fly-Over
$\boldsymbol{p}$	Parent, in Context of a State
$\boldsymbol{p}$	Partial, in Context of a Visibility Graph
$\boldsymbol{P}$	Planning
$\boldsymbol{pa}$	Previous Edge Angle
$\boldsymbol{pr}$	Propeller
$\boldsymbol{pt}$	Powertrain
$\boldsymbol{r}$	Reduced, in Context of a Visibility Graph
$\boldsymbol{r}$	Roll, in Context of Flight Dynamics
$\boldsymbol{rel}$	Relative
$\boldsymbol{ri}$	Roll In
$\boldsymbol{ro}$	Roll Out
$\boldsymbol{s}$	Start
$\boldsymbol{S}$	Stall

smp	Sample
ss	Steady State
t	Tangent
T	True Airspeed
ta	Tangent Edge Angle
tc	Tangent Candidate
to	Total
tp	Trajectory Planning
U	Update
wp	Weather Processing

Chapter 1

Introduction

The avoidance of adverse weather is an ever-present and safety-relevant task in manned and unmanned aviation. Approximately every fifth accident in commercial aviation and every fourth in general aviation is related to adverse weather [1, 2]. In particular thunderstorms and its surroundings are dangerous, as turbulence, gusts, wind shear, lightning, hail and icing may occur. How elaborate weather avoidance can be, is best illustrated by High Altitude Pseudo-Satellites (HAPS), which are a class of especially vulnerable fixed-wing aircraft [3]. Their relatively low weight and low performance increase the vulnerability to adverse weather [4]. So far the operation of these aircraft requires a considerable amount of manual effort in mission planning and execution. A large part of the incurring tasks are related to tactical flight planning based on external meteorological information, as some aircraft are not equipped with an onboard weather radar. For this purpose, pilots and meteorologists have to consider a large amount of information all at once, for example actual and future aircraft states, airspace restrictions and time-variant forecasts under consideration of their uncertainty. However, memory capacity of humans is usually limited to 7 ± 2 chunks of information [5], which is clearly insufficient to make optimal use of the available information. In complex scenarios with moving obstacles this shortcoming is partially compensated by applying quasi-static planning: Moving obstacles are avoided by an imaginary static radius of action plus an additional margin to account for uncertainty. This method works well when the aircraft moves fast in relation to the obstacles. However, convective weather has a high rate of change and quasi-static avoidance is prone to causing reactive flight guidance. Reactive trajectories are suboptimal in terms of safety, mission accomplishment and energy consumption. For unmanned aircraft a loss of link in both line-of-sight and beyond-line-of-sight communication is a potential incident. The ability to continue flight in these situations, at least for a short

period of time, enhances their level of autonomy and safety. The intrinsic complexity of anticipatory trajectory planning in uncertain dynamic environments calls for an automated solution. The consideration of kinematic and dynamic constraints further raise the complexity of the motion planning task [6, 7]. The kinematic constraints are to avoid time-varying thunderstorms, above a specified radar reflectivity threshold, while staying inside a restricted mission area. The exact solution of this kind of planning problem is considered to be NP-hard [8]. To overcome this hardness and find a near-optimal trajectory in reasonable time, the complex avoidance task is simplified to a geometric problem and is solved by combinatorial motion planning, for which comprehensive summaries can be found in [9, 10].

1.1 Objectives

The following list contains the objectives of this thesis. Several boundary conditions make the present planning task very demanding.

1. **Feasible Anticipatory Avoidance of Time-varying Thunderstorms**
   The main objective of this thesis is the development of an automated motion planning algorithm for unmanned aircraft, which computes feasible trajectories that presciently avoid obstacles with time-varying shape and position, e.g. thunderstorms. A collision-free trajectory, which is a path parameterized by time, has to lead from a start state to a goal state. Since the aircraft's velocity and its rate of change are finite and constrained, in both lateral and vertical direction (nonholonomic system), a planned trajectory has to be curvature-constrained.
2. **Fast, Reliable Computation of Near-optimal and Reproducible Results**
   Planned trajectories have to be optimal or near-optimal with respect to the belief about the future weather situation. It is intended that results are reproducible, as this is advantageous regarding a possible certification. This requires a deterministic property of the planner. The planning algorithm is intended for online-application, which implies that solutions have to be found in a short amount of time. For this purpose, a safety guarantee or probability that the planner's worst case execution time (WCET) complies with a permissible limit value has to be determined. Furthermore, reliability and a high success rate are essential. In case that no trajectory can be computed, a fallback planning solution is required, to ensure the aircraft's ability to act and to perform a safe flight at all times.

3. **Consideration of Uncertainty in External Environmental Information**
   The planner has to be able to plan safe trajectories solely relying on information about actual and future obstacles, provided by an external source, i.e. thunderstorm nowcast with a coarse update rate of five minutes. The external prediction regarding position and shape of thunderstorms is subject to considerable uncertainty, which has to be regarded for motion planning.

4. **Consideration of a Mission Area and Prescribed Safety Margins**
   For both civil and military operations, it is equally important that an aircraft stays exclusively in an assigned mission area and avoids, for example densely populated regions, restricted areas or national territory. Therefore, computed trajectories have to be exclusively located within a permitted area. Furthermore, they have to comply at any time with the recommended lateral clearance to thunderstorms, for example given by the Federal Aviation Administration.

5. **Verification of the Motion Planning Algorithm**
   Finally, the basic applicability of the presented planner to the weather avoidance task has to be verified.

1.2 State of the Art

In the context of aviation, weather avoidance and conflict resolution with other aircraft frequently go hand in hand. Many approaches are aimed at the automation of air traffic control and management tasks. In most cases weather avoidance is the task to circumnavigate thunderstorms, represented by *static* obstacles [11, 12, 13]. However, the interest is increasingly shifting towards the consideration of *time-variant* and uncertain adverse weather. An intermediate step is presented in [14], where a nonlinear model predictive control algorithm is presented, which solves conflicts between aircraft through convective weather by computing sets of locally optimal trajectories. Although the thunderstorms are static, the dynamic of the weather is considered by recurrent replanning. This approach is further developed in [15] by considering dynamic thunderstorm nowcast, which is represented by time-varying ellipses. A hierarchical approach with a receding horizon framework is presented, which uses an optimal control formulation. A high-level planner, which uses a low-fidelity aircraft model, regularly computes locally optimal trajectories, which simultaneously avoid time-varying obstacles and other aircraft. In [16] a stochastic optimal control approach for motion planning amid time-varying thunderstorms can be

found. A finite-horizon reach-avoid problem formulation [17] is applied to maximize the probability for one aircraft of reaching a given point, while avoiding stochastic obstacles under the consideration of uncertainties. Due to the curse of dimensionality [18] this kind of approach is unsuitable for fast computation of problems with high degrees of freedom. In summary, in the subject area of weather avoidance obstacles are frequently treated as static and their spatial extension is large to very large.

In contrast, in the extensively researched field of sense and avoid (SAA), obstacles typically represent other aircraft, which are dynamic but small in size and invariant in shape. The considered planning scopes are shorter than for weather avoidance and the focus lies mostly on reactive avoidance. Comprehensive summaries on this topic can be found in [19, 20]. The components of a SAA system are sensing, conflict detection, collision avoidance and flight control. In typical scenarios the runtime for the computation of a feasible trajectory is particularly critical. For the purpose of collision avoidance commonly geometric [21, 22], potential field [23, 24], sampling-based [25, 26] and numerical optimization [27, 28] approaches are applied [20]. Some methods, which are suitable for motion planning amidst moving obstacles, are discussed later in this section. At this point it is noted that in the field of weather avoidance, research is mostly carried out with static and large obstacles, while for collision avoidance the obstacles are mainly dynamic and small. This thesis fills a scientific gap, as it examines the scarcely researched avoidance of large dynamic obstacles.

The states of a nonholonomic system depend on the path taken to achieve them [29]. As a fixed-wing aircraft is an underactuated system, whose differential constraints are not analytically integrable, it is a nonholonomic system [10]. Its course change in turning-flight is proportional to the velocity (time derivative of the position). Therefore, it is not trivial to plan an optimal sequence of motion between configurations. While the rate of course change is constrained the space of possible configurations is not. This can be seen from the fact that a fixed-wing aircraft can reach every position in space, although it is not able to fly directly up- or sidewards. Considering nonholonomic constraints raises the complexity of motion planning [7] and is referred to as nonholonomic planning. If in addition the acceleration of a system is bounded, motion planning is far more complicated [10]. Problems which include differential constraints on the configuration, e.g. bounds on velocity, acceleration and external forces, are referred to kinodynamic planning [7]. According to [9] a kinodynamic motion planning problem is a nonholonomic motion planning problem, where a trajectory within a domain is computed that minimizes a scalar cost function, i.e. the total time to move from an initial state to a goal state, and is subject to

bounds on the allowed acceleration and velocity along the path. As a direct consequence, a solution has to be computed considering time as dimension. In the so called 2D asteroid avoidance problem, a collision-free trajectory for a point has to be determined, which moves in a plane with bounded speed. Obstacles are represented by convex polygons, which can exclusively move at constant velocity and do not collide. The exact solution of this problem is considered to be NP-hard [8].

Motion planning in real dynamic environments is generally subject to uncertainty in state sensing, state predictability, environment sensing and environment predictability [30]. This uncertainty can be considered explicitly, by applying bounded or probabilistic uncertainty, and implicitly via closed loop feedback strategies [10]. As dynamic environments do constantly change, associated information gets outdated. This imposes a real-time limitation on the motion planning, which is called decision time constraint. In a highly dynamical environment it may not be possible to find a complete or optimal trajectory in an allotted time interval, in which case partial planning can be applied [31].

The avoidance of adverse weather is basically a motion planning problem in the presence of time-varying obstacles, for which encompassing information is found in [10]. Moving obstacles entail several constraints for motion planning. Regarding kinematics they constrain the set of possible states (configurations at a certain point in time). In order to compute an optimal avoidance, instead of a geometric path, a trajectory has to be computed, which is a sequence of coordinates parameterized by time [10, 32].

For this purpose, the concept of configuration space and state space can be applied [10]. The configuration q of an aircraft is a combination of its degrees of freedom that for example can be composed of x-, y-position and course χ, which gives $q = (x, y, \chi)$. The configuration space C, with $q \in C$, is the set of all possible configurations or rather transformations that can be applied [10]. The concept was introduced by [33, 34] and is widely used for path planning in static environments. It enables the solution of very different planning problems using the same algorithms [10]. When dealing with moving obstacles, reactive planners use the static C-space to perform replanning whenever the environment changes [35, 36]. For anticipatory trajectory planning in the presence of moving obstacles, time has to be added to the C-space, which is called state space (X-space). Even two dimensional motion planning problems are difficult to solve as the dimensions are raised from two to three [9]. A trajectory in this case is a path, that is parameterized by monotonically increasing time and is compliant with kinematic and differential constraints of the aircraft. Time-dependent configurations are called states x, for example $\mathrm{x} = (x, y, \chi, t)$ with $\mathrm{x} \in X$ [10]. To compute optimal trajectories in

the presence of time-varying obstacles, under consideration of nonholonomic constraints, motion planning has to be performed in the state space X. Based on a knowledge or an estimate about the future obstacles, a sequence of states leading from start state to goal state can be determined. Different motion planning methods that are able to deal with moving obstacles are presented in the following.

For combinatorial motion planning the free search space is explicitly represented by a connectivity graph/roadmap in a first step, which can be very costly. The discrete topology, for example visibility graph, Voronoi diagram, exact or approximate cell decomposition, is subsequently searched for an optimal solution. In most exact planning problems the time complexity is exponentially related with the dimensions of the configuration space [37]. The upper bound for the complexity of exact motion planning in a three-dimensional dynamic environment is given by a general algorithm, which uses exact cell decomposition, with a time complexity that is twice exponential in the dimension of the configuration space [38]. Canny [6] later introduced a roadmap algorithm where time complexity is singly exponential in its configuration space dimension. Generally, the explicit representation of the search space limits the application of combinatorial methods to problems with low degrees of freedom, i.e. the number of necessary variables to specify x [10]. Nonetheless, combinatorial planning offers desirable properties like optimality, completeness and repeatable results which can be important for certification. In [39] a practical combinatorial planning method to compute time-optimal trajectories in the presence of moving obstacles is presented that uses a visibility graph representation of the search space. The algorithm is furthermore capable to deal with a moving goal. In [40] a heuristic search algorithm is presented, which computes safe time-minimal trajectories amidst unpredictably moving obstacles, i.e. growing discs.

Sampling-based motion planning algorithms are extremely popular due to their ability to quickly find feasible trajectories in X-space, for problems with high degrees of freedom and complicated constraints [41]. LaValle created the basis for effective sampling-based motion planning by introducing the rapidly-exploring random tree algorithm (RRT) in [42]. The fundamental idea is to avoid the aforementioned explicit representation of the configuration space, by using random samples. Rooted in a start node, the algorithm incrementally grows a tree of feasible trajectories, until a node is in a specified goal region. A drawback of RRT is that the existing node sequence is not updated, even if between start to goal a better one exists. This shortcoming was addressed by [43], who introduced RRT*. There are numerous further developments, for which a comprehensive summary can be found in [44]. Examples for kinodynamic planning in the presence of moving obstacles can

be found in [45, 46, 47]. In general, sampling-based methods have difficulties with narrow passages, as the likelihood for a sample being in free space decreases. Furthermore, if no system simulation module (local planner) is used, trajectories tend to be jagged and therefore require postprocessing, e.g. using B-splines, Bézier curves or clothoids [44]. If random samples are used, algorithms are probabilistically complete. This means that if a trajectory exists and the number of samples approaches infinity, the probability to find a trajectory converges to one.

Finite difference motion planning is also capable to deal with moving obstacles. The fast marching method (FMM) allows an approximate solution of the isochronous level sets, using a finite-difference method, which can be solved in a fast manner in two or three spacial dimensions, using the eikonal equation by [48] that accordingly to the Fermat's principle describes the shortest path between two points separated by optical media. In [49] a finite difference motion planning approach is introduced. A front propagates amidst moving obstacles traveling with bounded velocity. A trajectory is generated from an initial state to the goal state by using the vector field of the expansion wave and the state space equation of the system. The time-optimal trajectory is determined by backtracking. Compared to analytic solutions the discretization in the standard FMM leads to travel-time errors, which can be substantially reduced by applying a multistencil method [50]. Nonholonomic constraints cannot be considered directly, however, they can be modeled by penalizing curvatures, as proposed by [51]. The FMM is resolution-complete, near-optimal and generates smooth trajectories. Furthermore the anisotropic FMM is able to consider vector fields, for example wind.

1.3 Research Contributions

In this thesis, a deterministic combinatorial motion planner, called model predictive trajectory planner (MPTP), is introduced. Partial results of this work have been published in [52]. Under the assumption that the velocity and angle of climb of the aircraft are at least piecewise constant, collision-free, time-monotone, near-time-optimal and curvature-constrained trajectories from start x_s to goal state x_g are computed. Global optimization is performed by a heuristic search, i.e. A*-search algorithm [53]. For fast and reliable convergence of the planner, the X-space is iteratively built and its growth is curbed by a novel heuristic function, which effectively reduces the depth of search.

In the presented setup, motion planning is based on external environmental prediction, i.e. thunderstorm nowcast. Motion planning in the real-world is generally subject

to uncertainty in environment predictability [10]. This immanent uncertainty is explicitly modeled by the prediction unit, using different types of discrete margins. Based on the latest information available, the MPTP plans anticipatory trajectories, avoiding arbitrary moving obstacles, by iterating between its prediction and optimization unit. The aforementioned uncertainty is furthermore implicitly considered by a closed loop policy, similar to a model predictive controller. Recurrent replanning in regular intervals enables a reactive avoidance. Therefore, the MPTP concept combines anticipatory and reactive planning, which improves the chances for success even in environments with considerable uncertainty. The research contributions of this thesis are summarized subsequently.

1. **Estimated Future Conflict Areas and Estimated State Space** (Section 3.3)
   Using the fundamental concept of configuration space [33, 34], obstacles are transformed to an obstacle region, which is a set of configurations, where the system is in collision. This can be done by explicit sampling, which is an established method for the avoidance of obstacles [37]. Sampling-based planners use deterministic or random samples in conjunction with a collision detection to determine invalid configurations in C-space and invalid states in X-space [42, 41].
   Contribution: Exploiting the property that a shortest trajectory will always pass tangentially by an obstacle, this thesis demonstrates that it is safe and sufficient to identify radial state samples, propagated from an initial state, which are in collision with moving obstacles. Time-varying thunderstorms are transformed to static obstacles by superposition of estimated future aircraft states (from an initial state) with their contemporaneous prediction (including uncertainty). The generation of a state space by superposition of motion estimates, under consideration of uncertainty, has not yet been implemented in this form. A set of state samples, in which the aircraft is estimated to be in collision with thunderstorms, is called estimated conflict area (ECA). The free estimated state space (EX_{free}) is the difference of a workspace (mission area) and the set of estimated conflict areas. The assessment of future conflicts from an egocentric perspective, provides the basis for a fast and reliable computation of anticipatory trajectories in dynamic environments, using a combinatorial motion planning approach. As the problem space is reduced by the temporal dimension, moving obstacles can be avoided by solving a set of geometric problems.

2. **Matrix Computation of Visible Tangent and Bitangent Edges** (Section 4.1.2)
   The determination of tangent edges, between a point and vertices of polygons, and

bitangent edges, between vertices of polygons, for the construction of a reduced visibility graph was investigated by several authors including [54, 55, 10].

Contribution: A novel method for the computation of tangent and bitangent edges is introduced that exclusively uses matrix operations and logical indexing, which can reduce execution times considerably. At the same time, the convex vertices of obstacles are determined, leading to a new and less concave form of the obstacles and consequently to a lower number of edges for intersection tests. Furthermore, a new method is introduced which allows to determine the visibility of the aforementioned edges, solely by counting the number of their intersections with obstacle edges. By using information from the matrices, which are used to identify tangent and bitangent edges, the method is able to deal with geometric degeneracies, i.e. collinearity. In this way, a reduced visibility graph VG_r can be computed. Investigations regarding the branching factor (mean number of connected vertices) of reduced and unreduced visibility graphs, exhibit a considerable advantage regarding a subsequent A*-search.

3. **Partial Shortest Path Map** (Section 4.1.2, 4.3.1 and 4.3.2)

   A roadmap in form of a visibility graph can be searched in order to compute a shortest path. However, the construction of a complete visibility graph has a lower bound in big Omega notation [56] (lower bound for the time-complexity of a function) of $\Omega(|V|^2)$ or $\Omega(|E|)$, where $|V|$ is the total number of obstacle vertices and $|E|$ is the resulting number of edges in the visibility graph [57]. To speed up the search for a shortest path, the idea is to omit the construction of a separate visibility graph and to build a partial shortest path map (SPM) instead [9]. The continuous Dijkstra paradigm by [58] is an effective approach to compute a SPM that runs in $O(|V| \log |V|)$. However, existing sophisticated algorithms by [59, 60] are difficult to implement, due to complex data structures and techniques [61]. The fast marching method (FMM) [62], which is closely related to Dijkstra's algorithm and runs in $O(|V| \log |V|)$, can also be applied.

   Contribution: If, as in the present case, solely the shortest path to one point is to be generated, even more computational effort can be saved by computing the introduced partial shortest path map (PSPM). While both the continuous Dijkstra and FMM visit the complete environment, the presented approach uses exclusively visible tangent edges in combination with an informed search algorithm, i.e. A* with Euclidean distance heuristic, whereby the expansion of the search is automatically

curbed. A tree rooted in the start vertex is iteratively built, which results in a partial shortest path map, amidst static obstacles, or partial shortest trajectory map (PSTM), amidst moving obstacles. The presented algorithm for the computation of visible tangent edges runs in $O(|V| \log |V|)$ and space of $O(|V|)$, where $|V|$ is the number of vertices in the free estimated state space. This is repeated until a visible connection between start and goal is established. The search is complete and finds the shortest path or trajectory, if a consistent heuristic function is applied.

4. **A*-Search in Dynamic Environments - Partial Shortest Trajectory Map** (Section 4.2.2)

   For planning in partially known environment with moving obstacles, the A*-search is applied in combination with a replanning policy, which means that the search is performed on static obstacles in C-space, and their motion is considered by replanning, e.g. focussed D* algorithm [35] and anytime A* for dynamic environment [63]. The A*-search has been applied for planning in the presence of moving obstacles, for example in [40].

   Contribution: Here, an iterative A*-search in estimated state spaces (EX) is proposed, until the visibility from start to goal state is established, which results in a partial shortest trajectory map (PSTM) of the dynamic environment. Furthermore, the omission of certain queries in the A*-search algorithm is proposed for the special case of a completely dynamic environment (no static obstacles), which can improve the runtimes, at a low risk of sacrificing optimality. Correctness and optimality of the presented approach are demonstrated, by comparing the results with Dijkstra's algorithm, which is guaranteed to be optimal. In order to consider nonholonomic constraints, the open list of the A*-search stores various state information, for example course angle, bank angle and time of arrival, which are used to determine feasible/reachable states.

5. **Heuristic for A*-Search in Dynamic Environments** (Section 4.3.3)

   In order to assess the total cost from start to goal, the A*-search requires a heuristic estimate about the actual cost to the goal. One of the most commonly applied heuristics is the Euclidean distance, which neglects obstacles and is particularly unsuitable if they are large, dynamic and concave.

   Contribution: In order to enable fast combinatorial planning in state space X, a novel and particularly targeted heuristic, called Shortest Static Path Heuristic (SSPH), is introduced. It estimates the cost-to-go of states which are adjacent to

an initial state, in the estimated state space of the initial state. A nested A*-search can be applied, to either search a reduced visibility graph VG_r or to build a shortest static path, to compute the heuristic cost. This generally limits the search effort efficiently and opens up the possibility to use the presented motion planner in real-time. To the best knowledge of the author this approach does not exist in the literature. Although SSPH is inadmissible due to a possible overestimation of the cost-to-go, results of a Monte Carlo simulation prove, that computed trajectories are mostly near-optimal, while the runtimes are greatly improved, compared to the optimal methods, i.e. A*-search using Euclidean distance heuristic and Dijkstra's algorithm.

6. **Methods for the Computation of Feasible States** (Section 5.1 and 5.3)
   The forward simulation of reachable states is typical for sampling-based motion planning approaches under differential constraints [42]. A system simulation module computes feasible state transitions from an initial state by integration, for example using Euler method or Runge-Kutta methods. If a simulated state does not already exist and not collide with an obstacle, it is added to a search graph [10]. The approach in this thesis applies a similar procedure to combinatorial planning. Nonholonomic constraints of the aircraft are considered by adding so-called auxiliary states, which are dynamically feasible states from an initial state, to the search graph.
   Contribution: For the determination of auxiliary states, different methods for low- and high-fidelity representations of aircraft dynamics are introduced. As states are waypoints parameterized by time, it is important to asses their feasibility. Similar to the standard estimations of static waypoints in [64], this is accomplished by geometric considerations. This novel method is a fast and simple way to evaluate both feasibility and mean stabilization distance of fly-by and fly-over states (waypoints at a given time). Furthermore, for an improved guidance of low-performance aircraft, feasible states are simulated by a fast noniterative method, which is based on the Taylor series and analytically computes reinitialization times in order not to exceed the commanded course change or maximum bank angle. Thus, the performance of the aircraft is considered at a time-scale which is below the time increment of the trajectory planner.

7. **Implicit Planning of Holding Patterns** (Section 5.4)
   The possibility exists that the goal is temporarily inaccessible in the planning hori-

zon. This is all the more likely if an explicit representation of uncertainty is applied, as this can lead to excessive coverage of free space. Therefore, it is important that a planner is capable to employ authorized planning maneuvers, until the goal is cleared.

Contribution: Due to the addition of auxiliary states, the presented MPTP has the immanent ability to compute trajectories, which include authorized holding patterns, for example in case that the goal state is covered at some time by a thunderstorm. By constraining the turning-sense of the aircraft, the planner automatically generates holdings in form of circular patterns, which are an approved maneuver for unmanned aircraft. This ability greatly increases the likelihood that a feasible and sensible trajectory can be computed. It is common for motion planning approaches under differential constraints to add feasible states to the search graph [42, 10]. However, to the best knowledge of the author, these states are not used for the planning of holding patterns.

Besides the obvious application for autonomous avoidance of moving obstacles, the presented planner can also be used as initial guess generator, for example for gradient-based trajectory optimization in dynamic environments [10]. Thus, an optimal control sequence in the homotopy class of the initial guess can be determined, using a high-fidelity aircraft model [65, 66]. It can be furthermore applied to all kinds of nonholonomic and kinodynamic planning problems, with nonlinearly moving obstacles, whose motion cannot be described analytically.

1.4 Structure of the Thesis

Throughout this thesis, the results of individual components are presented in their respective chapter, whereas the overall results of the trajectory planner are found in Section 6.

The thesis is organized as follows. In Section 2, the functional principle of the presented MPTP is introduced. The prediction unit of the planner and its models are described in Section 3. In Section 4, the search space representation is introduced. Furthermore, methods to improve time and space complexity requirements are discussed, followed by a comparison of different visibility graphs and the implications, regarding the ensuing graph search. At the end of Section 4 an informed search algorithm is presented, alongside with different heuristics. Section 5 features three methods for the estimation and simulation of feasible states, and explains the applied methodology, to compute curvature-constrained

trajectories. Subsequently, the automatic planning of holding patterns is introduced. Section 6 presents simulation results for the key capabilities of the MPTP including a reliability assessment by a Monte Carlo simulation, followed by a discussion in Section 7. Finally, the thesis is completed by the conclusion in Section 8.

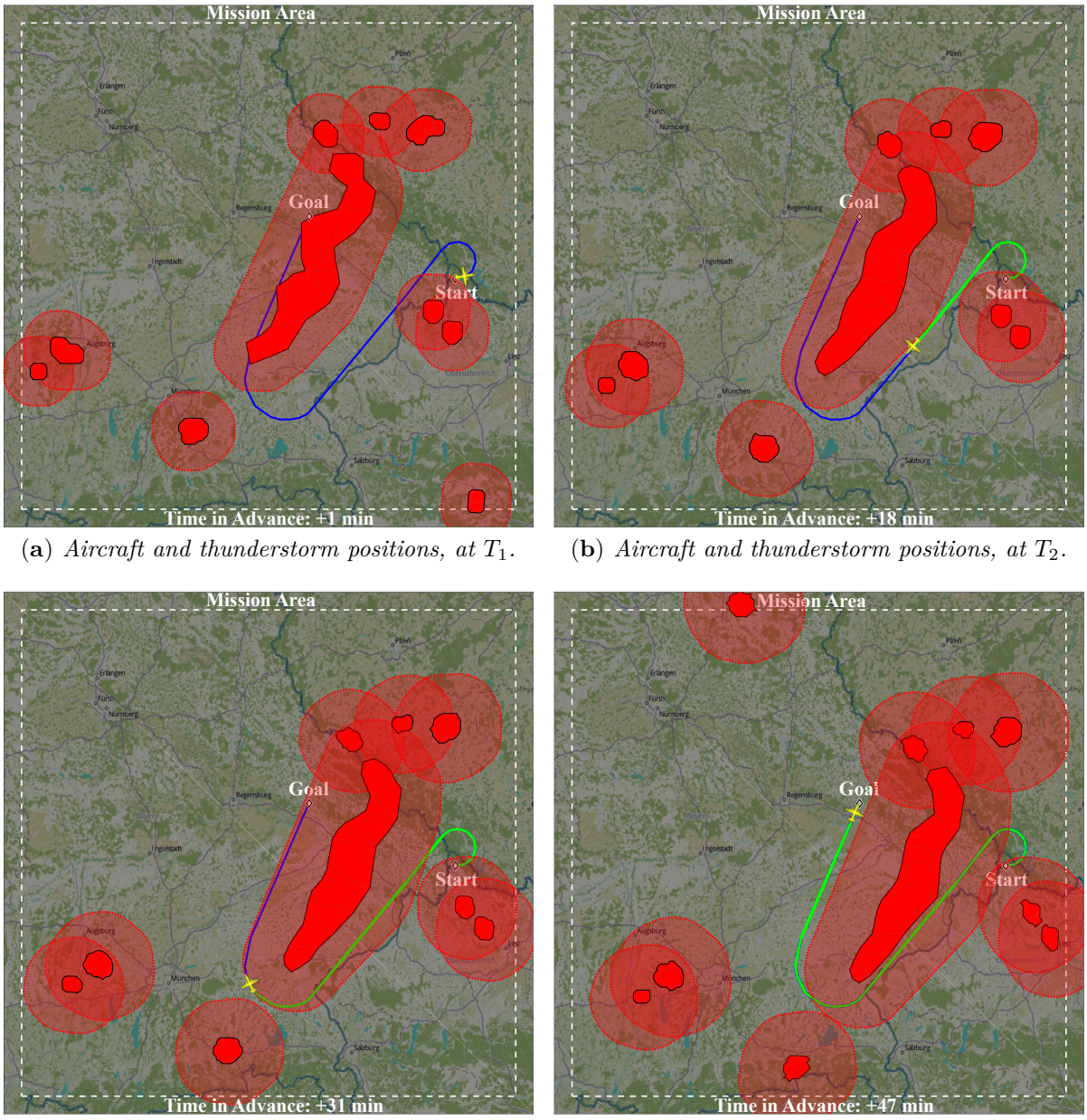
Chapter 2

Model Predictive Trajectory Planner

In this chapter, the concept of an anticipatory trajectory planner is presented. Planning is performed by a reactive guidance loop, which resembles the setup of a model predictive controller (MPC), due to the recurring feedback of environmental and state information. It is therefore termed Model Predictive Trajectory Planner (MPTP) and provides, as it will be shown in Chapter 6, resolution-complete, globally near-optimal and dynamically feasible trajectories for the expected case. Its individual components are explained in detail in the following Chapters 3, 4 and 5. The MPTP's task is to compute feasible trajectories from a start state x_s to a goal state x_g, avoiding time-varying and uncertain thunderstorms, and staying inside a prescribed mission area. As safety has the highest priority, an optimal trajectory is defined as the one, with the shortest distance between start and goal, while maintaining recommended clearance to thunderstorms and its surroundings at all times. Because it is difficult to determine the appropriate altitude for vertical avoidance of thunderstorms (visible top not necessarily equal to radar top) and turbulence is frequently encountered above storm clouds, the focus lies on lateral avoidance [67]. Some unmanned aircraft, like HAPS for example, are not equipped with onboard sensors, such as a weather radar. Thus, the anticipatory planning relies exclusively on an external prediction, i.e. thunderstorm nowcast. The development of the MPTP was decisively influenced by the following two principles:
"The philosophy of avoidance is an integral part of flight planning", by the Federal Aviation Administration [68] and "Simplicity is prerequisite for reliability", by Edsger W. Dijkstra.

2.1 Anticipatory Motion Planning

The memory capacity of humans is limited to approximately 7±2 chunks of information [5]. In complex environments with time-varying obstacles, this shortcoming can be partially compensated by applying quasi-static planning. In this case moving obstacles are avoided by an imaginary static radius of action and an additional margin, to account for uncertainty. This method can be successfully applied, if the aircraft moves relatively fast in relation to thunderstorms (obstacles). However, for low-performance aircraft this can lead to a reactive guidance, especially as convective weather has a high rate of change. This results in suboptimal trajectories, regarding safety, energy consumption and mission accomplishment. These undesirable effects can be avoided by applying anticipatory trajectory planning, for which Figure 2.1 shows an example. Furthermore, a loss of link, in both line-of-sight and beyond-line-of-sight communication, is a potential incident for unmanned aircraft, which cannot be excluded. The ability to continue the flight safely in this situation, at least for a limited period, could enhance their level of autonomy and safety considerably. The intrinsic complexity of anticipatory trajectory planning in uncertain dynamic environments calls for an automated solution. The reduction in workload, improved safety and mission accomplishment, resulting from the enhanced autonomy, are the motivation for this thesis. In order to be valid, a trajectory has to be compliant with numerous constraints. A fixed-wing aircraft is a nonholonomic system that requires nonholonomic motion planning. In order to compute curvature constrained trajectories, dynamically feasible states are added to the search graph, which is presented in Chapter 5 that is an extension of the optimization in Chapter 4. Generally problems, which include differential constraints on the configuration are referred to kinodynamic planning [7]. The velocity of the aircraft is assumed to be constant, from which follows that the acceleration is zero. Thus, the presented approach can be categorized as a simplified kinodynamic motion planning problem. The moving obstacles (throughout the rest of the thesis thunderstorms are treated as obstacles) are global kinematic constraints that limit the states of the aircraft [7]. For this purpose, time has to be considered, which is why the motion planning in Chapter 3 and 4 is directly performed in state space X [69, 10]. Generally, planning in a real dynamic environment is also subject to uncertainty, regarding recognition and prediction of the state and the environment [30]. The latter is especially true for the external thunderstorm information obtained from nowcasts (see Section 3.1), as thunderstorms cannot be predicted exactly due to their highly dynamic and partially random evolution. Since the presented planning algorithm is intended for



(**a**) *Aircraft and thunderstorm positions, at* T_1.

(**b**) *Aircraft and thunderstorm positions, at* T_2.

(**c**) *Aircraft and thunderstorm positions, at* T_3.

(**d**) *Aircraft and thunderstorm positions, at* T_4.

Figure 2.1: *An aircraft flies on an anticipatory trajectory, computed by the MPTP, at time* t_0. *The red areas represent thunderstorms, which are surrounded by growing discrete margins (transparent red areas). If the aircraft moves slow in relation to the thunderstorms, the compliance with safety margins is a demanding task.*

online application, an additional real-time decision constraint is required for the planner, due to environmental uncertainty. A fallback solution, e.g. partial planning [31], is required to ensure the aircraft's ability to act, in case that the planner does not converge in an allotted time interval. Figure 2.2 gives a graphical overview about the applying boundary conditions for the present aircraft motion planning problem and therein provides a classification of the presented MPTP.

Figure 2.2: *Summary of the boundary conditions for the present motion planning problem. The red lines indicate the applying conditions.*

2.2 Framework of the Model Predictive Trajectory Planner

The presented MPTP consists of a prediction and optimization unit, which is another resemblance to a MPC besides the feedback loop (Figure 2.3). The prediction unit (Chapter 3) processes the dynamics of both aircraft and environment, to generate an estimation of the free state space, which excludes sets of invalid states, i.e. estimated conflict areas. For the aircraft it is assumed, that both the angle of climb and velocity are at least piecewise constant. This is reasonable, as the en-route velocity band of an aircraft is generally narrow. The motion of thunderstorms is based on nowcast, which is subject to significant uncertainty. This uncertainty is implicitly considered by a closed feedback loop of the MPTP and explicitly modeled in the prediction unit, by discrete time-variable bounded margins (Section 3.1.1). Their size is a function of a probability and time, as proposed by [70]. The optimization unit creates a graph representation of the aforementioned free estimated state space and performs an informed search (see Chapter 4). In order to find a feasible trajectory in a dynamic environment, the MPTP iterates between these two units, until a solution is found or failure is reported. The drawback of the aforementioned explicit uncertainty representation can be an over-conservative coverage of free space. The immanent ability to automatically plan holding patterns (Section 5.4), increases the likelihood, that a trajectory can be computed, in case, that the goal state or the access to it is temporally covered. This reduces the number of cases, in which no solution can be found.

Avoidance basically consists of two goal-oriented tasks i.e. anticipatory planning and reactive evasion maneuvers. A hierarchical framework is presented in which the MPTP, which plans with a low-fidelity aircraft model at a low-update-rate, and the flight controller, which operates on the actual aircraft at a high-update-rate, are decoupled, as in [71, 72, 15]. The high-level MPTP recurrently plans from the actual state (initial state feedback by the aircraft, Figure 2.3) to the actual goal (reference by the mission planner, Figure 2.3), based on the latest nowcast (predicted disturbances, Figure 2.3). The periodic correction of control errors and environmental prediction yields robustness [71]. Its control output is a set of states, that is transmitted to the low-level flight controller, which acts as reactive component and ensures that the aircraft reaches commanded states in due time, compensating for unpredicted disturbances (see Figure 2.3). As mentioned in Section 2.1, the velocity of the fixed-wing aircraft is finite and nonholonomic constraints for turning

flight have to be considered, which is described in Chapter 5. Thereby, the feasibility of trajectories by the high-level planner can be ensured for the wind-free case. The resulting curvature-constrained trajectories do not necessarily require post-processing. Müller demonstrated the feasibility of the proposed guidance strategy, with the setup depicted in Figure 2.3, in [3]. The simulations are performed with a six degree of freedom aircraft

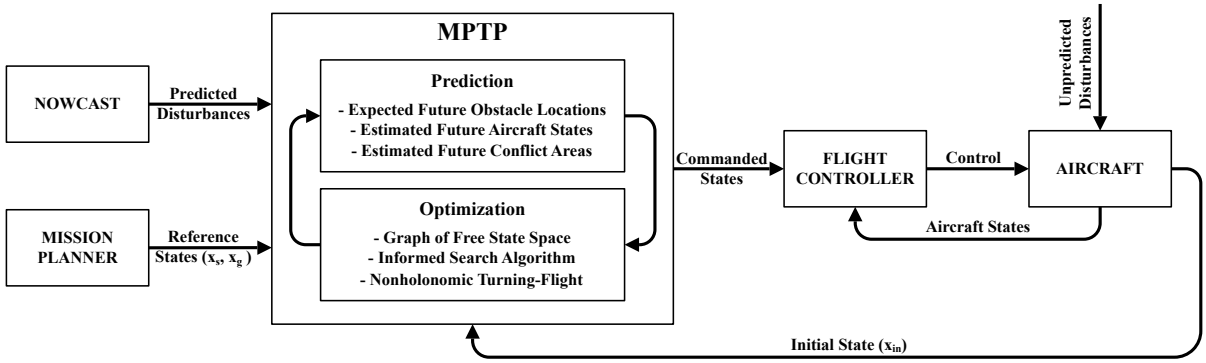


Figure 2.3: *Schematic of the presented MPTP, embedded in its environment.*

model, flight controller and historical wind data. The update horizon T_U is equal to the external nowcast update horizon time $\Delta T_N = 300$ s. Therefore, at least twelve trajectories are computed in the period of one hour. The sample time is Δt and the planning horizon T_P is identical to the nowcast horizon T_N of one hour. This is necessary in order to determine, if a trajectory to the goal state can be computed. The MPTP recurrently performs planning, which is triggered by every update of the nowcast. The latest weather data is immediately processed, which takes a variable runtime δt_{wp} that depends on the size of the weather data, the selected procedures (see Section 3.1) and usually needs only a few seconds. Figure 2.4 depicts the proposed cycle scheme of the MPTP. While planning

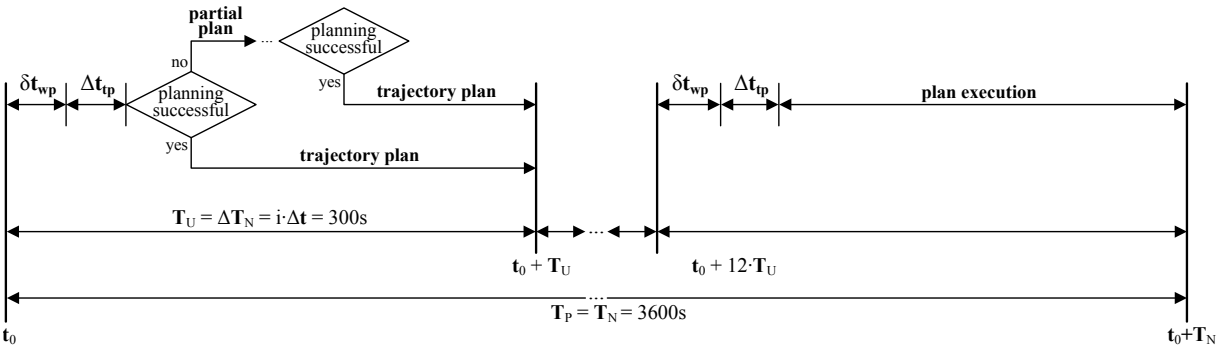


Figure 2.4: *One hour cycle scheme of the MPTP.*

takes place, information outdates in dynamic environments. This imposes a real-time decision constraint and limits the allotted time for the computation of a trajectory [31]. The variable trajectory planning runtime δt_{tp} is therefore limited to a fixed value Δt_{tp}, in

order to avoid inactivity [71, 73]. If $\delta t_{tp} < \Delta t_{tp}$, the plan execution starts immediately at $\delta t_{wp} + \delta t_{tp}$ after the nowcast update.

As convergence of the MPTP is not guaranteed, a fallback solution is necessary in order to ensure the aircraft's safety. Section 3.3 introduces a partial motion planning strategy, which procures the MPTP time to replan, while the aircraft moves on estimated safe states. Alternatively, holding patterns (see Section 5.4) can be performed in free state space, while replanning takes place.

Chapter 3

Prediction - Estimation of Conflicts

In this chapter the prediction models of the previously presented model predictive trajectory planner (MPTP) are described. The first model predicts the obstacles, i.e. the expected future thunderstorm areas. For this purpose the nowcast is interpreted regarding the relative hazard for a specific aircraft, taking into account the immanent uncertainty (Section 3.1). A second prediction model computes future aircraft state samples (Section 3.2). In order to estimate future conflicts (Section 3.3), the aforementioned models are finally superimposed. As the term conflict seems more appropriate than collision in the context of weather avoidance, it will be used synonymously throughout this thesis.

3.1 Expected Future Obstacle Locations

Thunderstorms and their surroundings are extremely dangerous for all kinds of aircraft. The expectation of their future location is based on thunderstorm nowcast, which in this case is historical weather data issued by the algorithm Radar Tracking and Monitoring (Rad-TRAM) [74]. The data is provided in Extensible Markup Language format (XML) by the German Aerospace Center DLR. An XML-file contains nowcasts as well as other measurements. Thunderstorms are represented as discrete polygons, which ensures small file size and allows fast processing [75]. Figure 3.1 shows the basic nowcast file structure. Green arrows indicate real time and the blue ones nowcast time. A nowcast has a temporal horizon T_N of one hour and is updated at an interval ΔT_N of five minutes. Therefore, each nowcast consists of thirteen sets $(t_0 + 0 \text{ min}, t_0 + 5 \text{ min}, \ldots, t_0 + 60 \text{ min})$, containing the coordinates of storm cells and additional information, for example geometric center, movement direction and speed. The first set contains the measured thunderstorm situation, while the others are predictions.

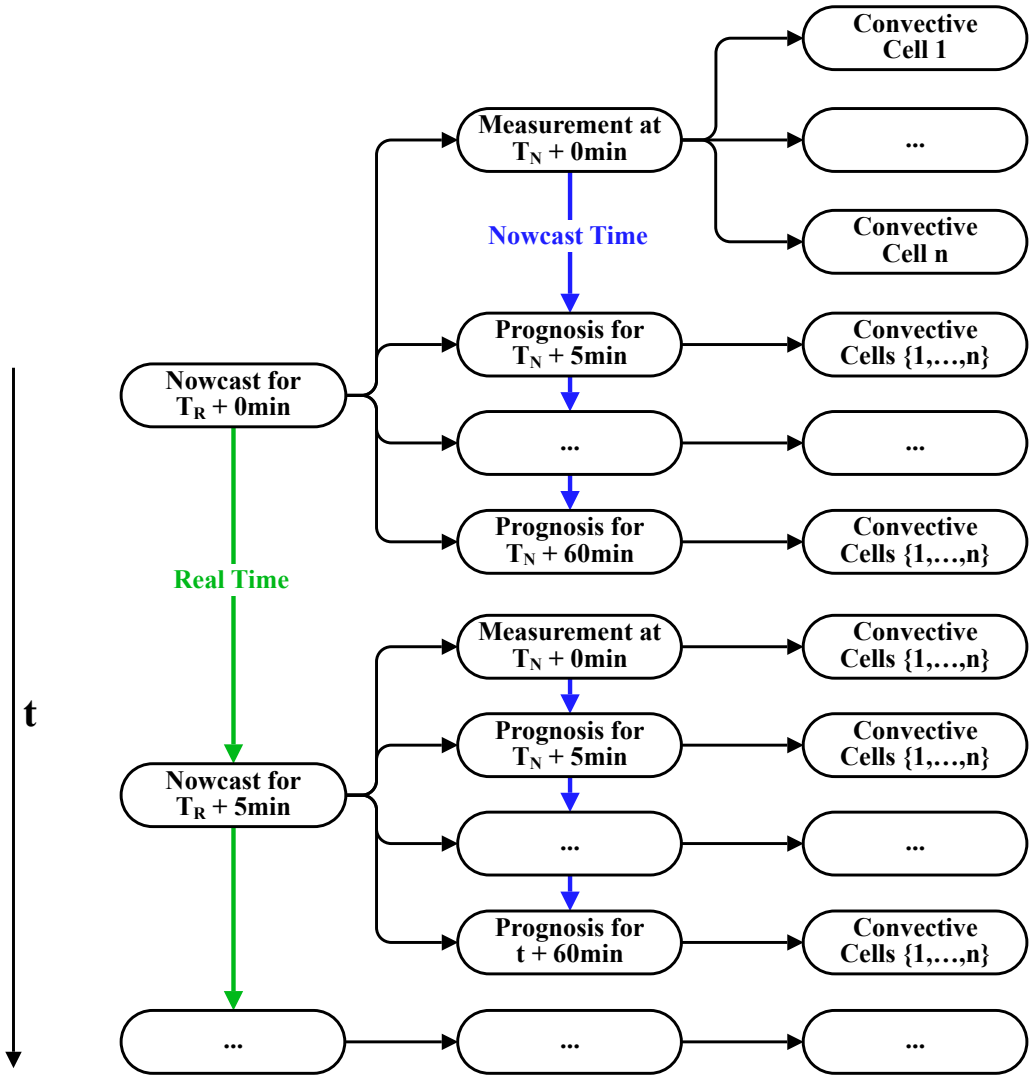


Figure 3.1: *Basic structure of a thunderstorm nowcast data set.*

In aviation a minimum lateral clearance to thunderstorms (especially cumulonimbus clouds) is recommended [67, 76]. In order to compute safe trajectories, the weather data has to be interpreted with respect to the limitations of an aircraft. For this purpose, storm cells are enlarged by appropriate margins. Two kinds of margins are applied to nowcasted cells, which are described in the following sections. Their purpose is to account for different kinds of uncertainty regarding the weather nowcast. Information about the wind field on standard flight levels is based on COSMO-DE model [77] data and supplied in General Regularly-distributed Information in Binary form (GRIB). The forecast horizon is 21 hours and the update interval is one hour [78].

3.1.1 Probabilistic Margins

Due to initial condition uncertainties and model errors, forecasts are never accurate [79]. Especially convective weather has a high rate of change and is difficult to predict. The forecast errors generally increase with advancing time. Here, this fact is explicitly modeled by applying bounded uncertainty margins. Thunderstorm cells are enlarged by time-varying, nonuniform deterministic margins (dark red line in Figure 3.2).

The construction of these probabilistic margins is based on the approach in [70]. In a first step, an original concave thunderstorm polygon is transformed to a convex polygon (see Figure 3.2, where the concave portion is indicated by dotted red line). This is reasonable, as concave radar signatures are associated with strong turbulence and hail [67]. Then, a coordinate system is placed in the geometric center (GC) of the convex polygon (light red polygon in Figure 3.2). The orientation of the abscissa is determined by the moving direction vector (bold black arrow). As shown in Figure 3.2, the offset

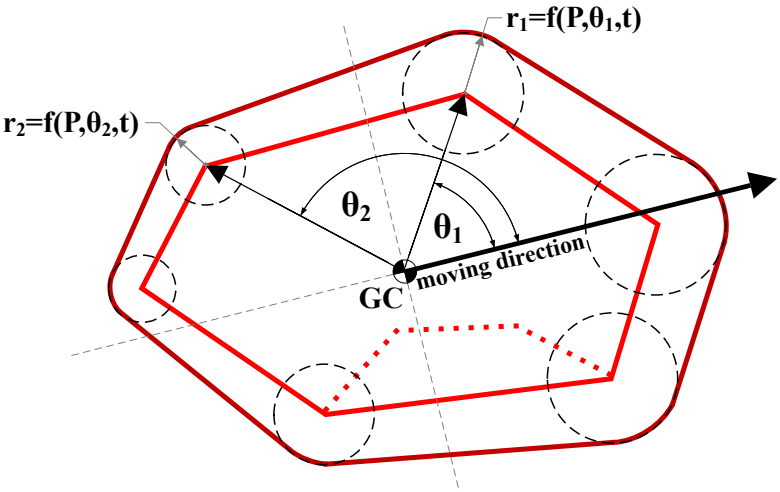


Figure 3.2: *Construction of a probabilistic margin based on [70]. It grows with time and is biggest in moving and smallest in the opposite direction.*

of the probabilistic margin to the thunderstorm polygon is constructed by circles with radius $r = f(P, \theta, t)$. P is the probability that a cell will be included by its corresponding margin at a given time in the nowcast horizon T_N, for which specific values can be found in [70]. $\theta \in [-\pi, \pi]$ is the angle between the abscissa and the vector from the GC to a vertex, and t is the time in the interval of t_0 to $t_0 + T_N$. The probabilistic margin is computed for each thunderstorm polygon and all discrete planning increments, from t_0 to $t_0 + T_N$ in intervals of Δt. Figure 3.3 shows the radii for two different probabilities P. The upper surface is for a high and the lower one for a low value of P. With advancing time t the margins grow. This is a safe procedure, especially when assuming high values

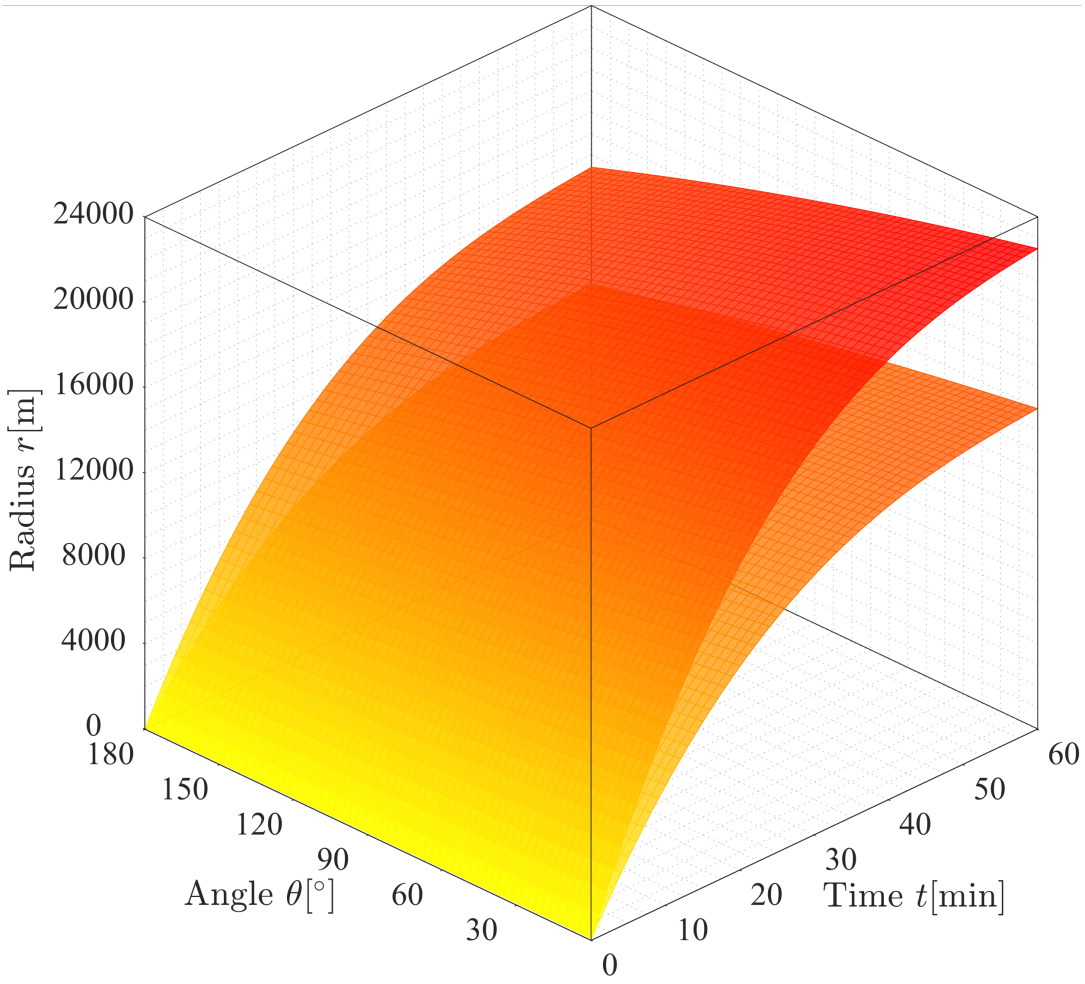


Figure 3.3: *Interpolated margin radius values (colormap) for the construction of exemplary probability margins, as function of time t, angle θ and a probability P.*

for P. As the presented MPTP computes trajectories based on the expected case, an overestimation of uncertainty leads to excessive coverage of free airspace. This directly affects the optimality of trajectories and sometimes even prevents that a solution can be found [10]. The ability to automatically plan holding patterns (Section 5.4) can solve some of the cases, in which the goal is covered or the access to the goal is blocked. The applied strategy is to compute trajectories in parallel using different probabilities P and then choosing the feasible solution, with the highest P or rather largest margins.

3.1.2 Safety Margins

In contrast to the aforementioned margins, the safety margins account for the uncertainty regarding the presence of thunderstorm accompanying phenomena, for example strong winds, wind shear, turbulence (e.g. clear air turbulence over and downwind of cumulonimbus clouds), lightning, icing and hail. These phenomena are hardly predictable

and often detected as late as when being in situ. This is why FAA recommends minimum clearances to thunderstorms [80, 76]. These safety margins are independent from the nowcast time. Their size partly depends on the structural limits of an aircraft and furthermore on appearance, size (area) and moving speed of a cell, as these parameters give an indication on the associated potential danger. In most cases it is safer to avoid thunderstorms on the upwind side as turbulence and hail is frequently encountered downwind [67]. By enlarging cells in these areas, the MPTP is tempted to shift the trajectory. Table 3.1 shows exemplary values for the aforementioned dependence of safety margins on external circumstances.

Table 3.1: *Exemplary values for situation dependent safety margins, by which thunderstorms should be avoided, as function of the appearance and wind speed, based on [81].*

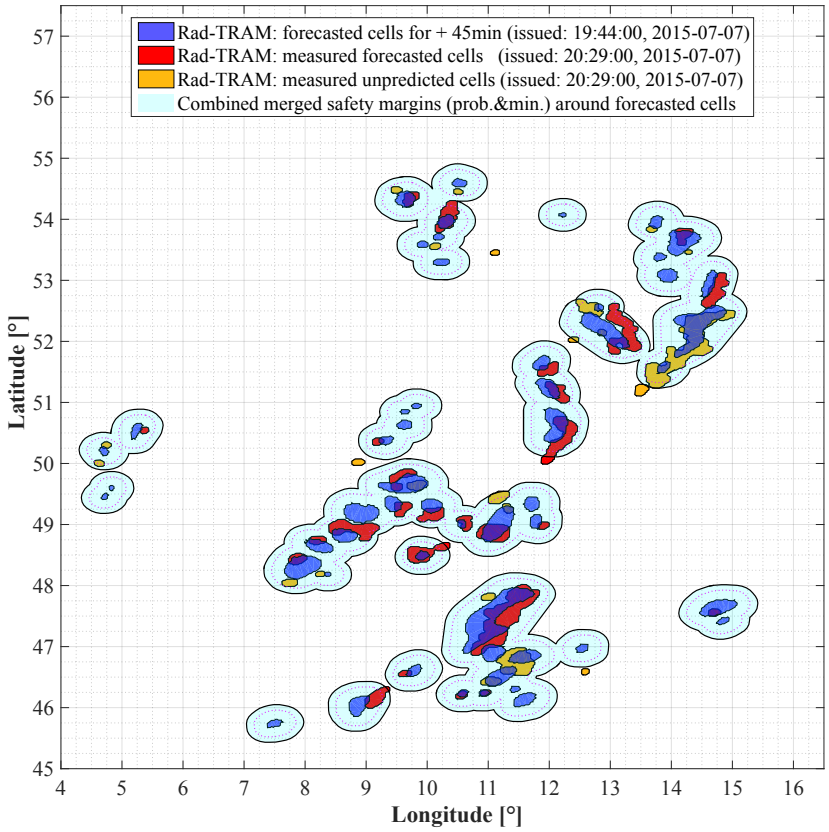
Wind Speed [m/s]	**Single Storm** [NM]	**Line of Storms** [NM]
< 10	20	30
≥ 10 to < 20	25	35
≥ 20	30	40

3.1.3 Merging of Margins

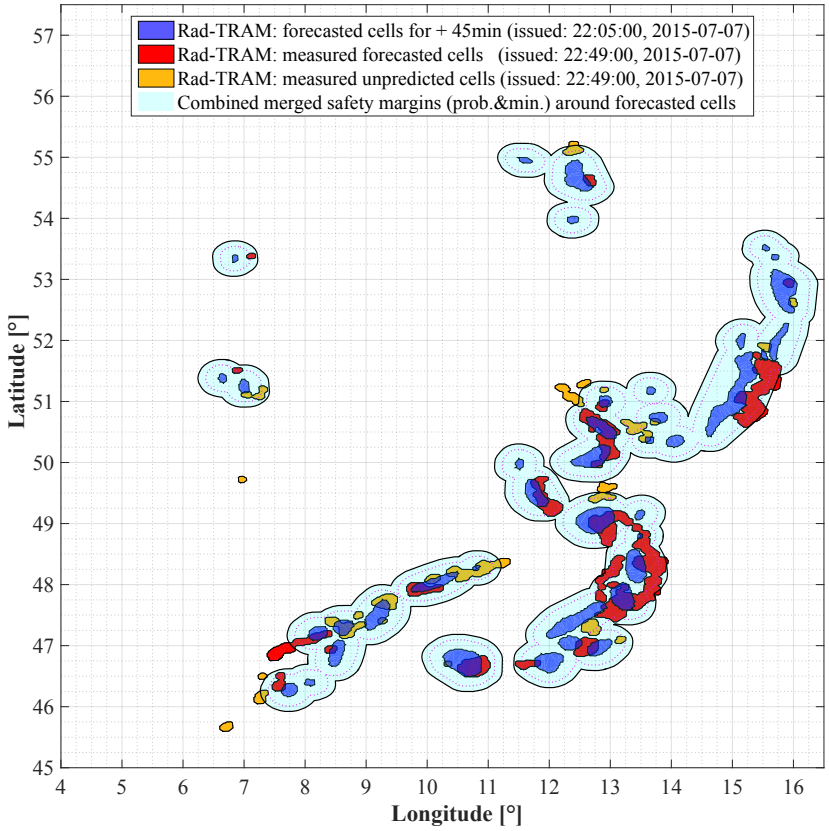
Safety margins and probabilistic margins can be applied individually or together, however their combination has previously not been investigated. Two examples for the superposition are depicted in Figure 3.4. The nowcast for $t_0 + 45$ minutes, for 07.07.2016 at 19:44 h and 22:05 h, are respectively depicted in blue. Merged margins are depicted in light blue color, while the subsequent true measurements of Rad-TRAM, for the prediction time, are depicted in red and yellow color. Heavy rain cells with the same identification number, as in the initial nowcast, are red, while measured cells with different identification numbers are shown in yellow and are either new or merged with other cells.

3.1.4 Clustering of Margins

High thunderstorm density and close neighboring cells indicate potentially weather active areas. These are dangerous because the uncertainty in the nowcast is locally high and sudden changes may happen. The FAA advises to circumnavigate weather active areas with a thunderstorm coverage of 6/10 or higher [76]. An intuitive interpretation is necessary in order to identify weather active areas, as planning trajectories through them can cause substantial short-term changes, with respect to the initial route.

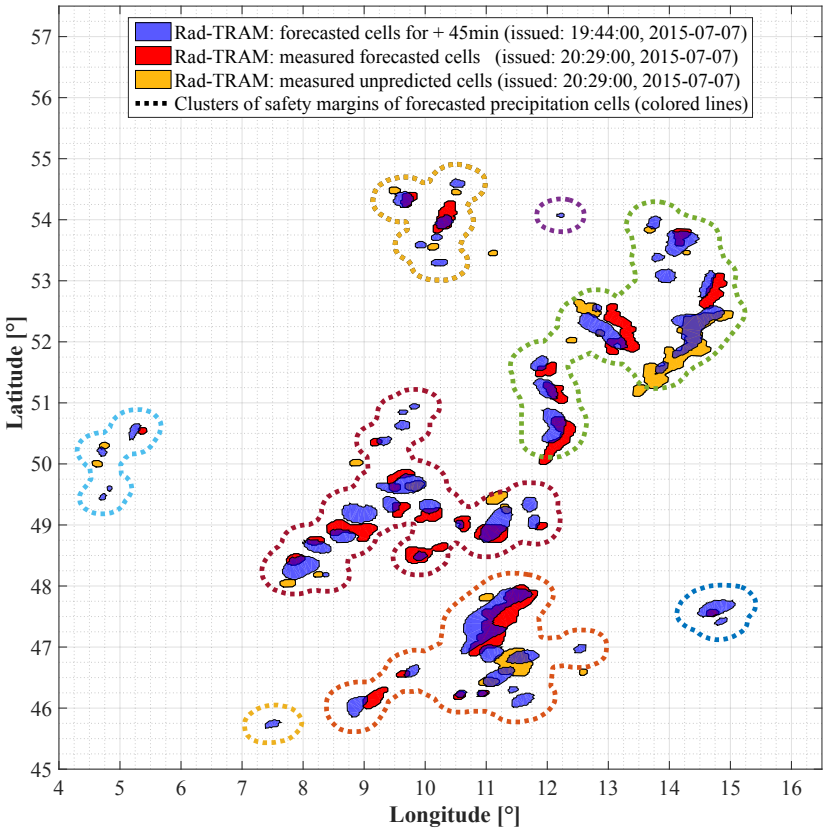


(**a**) *Superposition of margins, for 19:44h,*

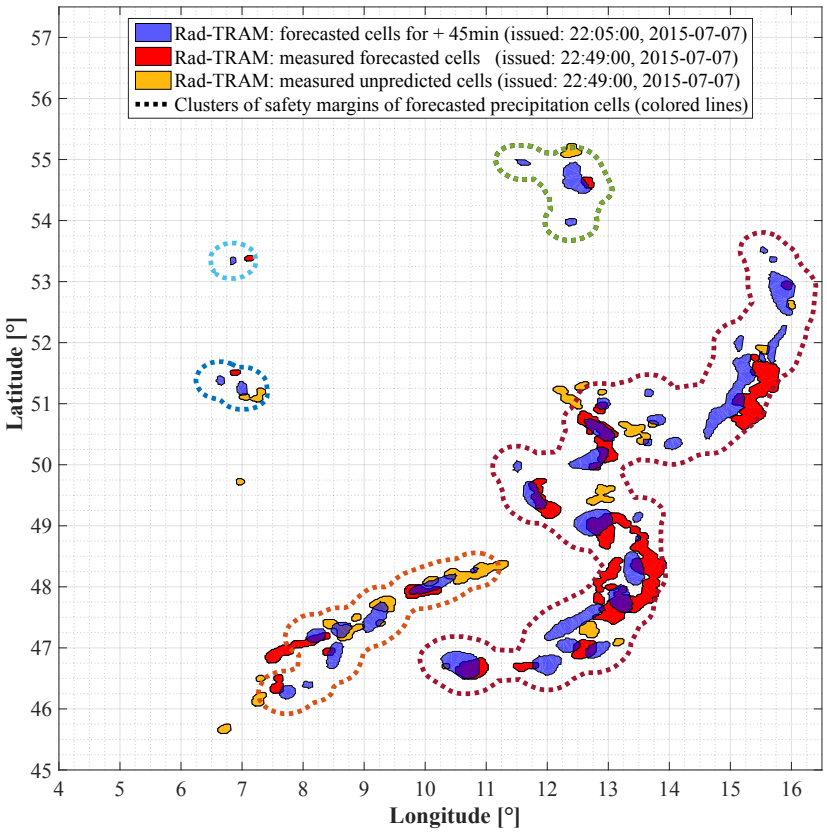


(**b**) *Superposition of margins, for 22:05 h.*

Figure 3.4: *Superposition of probabilistic and safety margins for thunderstorms on 07.07.2016 at (a) 19:44 h and (b) 22:05 h, for* $t_0 + 45$ *min.*



(**a**) *Clustered margins, for 19:44 h.*



(**b**) *Clustered margins, for 22:05 h.*

Figure 3.5: *Two examples for clustered margins illustrate the suitability for robust trajectory planning, as an intuitive interpretation is given. Despite using large margins some unpredicted storm cells are not covered.*

Discrete margins, as presented in the previous sections, have the disadvantage, that narrow passages may exist, which are likely to vanish in subsequent nowcasts. A human navigator intuitively avoids such areas, while a planner may uses them, when computing an optimal trajectory. Therefore, a function is required, which is able to deliver an interpretation of the situation. The cluster algorithm DBSCAN, which stands for Density-Based Spatial Clustering of Applications with Noise [82], is suitable for this task. A cluster is defined as a set of densely connected points. The main feature is the ability to detect contiguous areas of any shape and size. The main disadvantage is the sensitivity to the choice of parameters. However, by using the fixed mesh size this problem does not arise, since the density for the core points is constant. In order to generate a set of points a grid with a fixed mesh size 2×2 km is generated, which corresponds to the resolution of the radar data of the European radar ensemble, published by the German meteorological service (DWD), used for the Rad-TRAM nowcast [74]. The grid points, which are covered by the margins around thunderstorms, are selected as candidate points for the cluster analysis.

The parameters to detect a cluster are a maximum distance $\epsilon_{np} \in \mathbb{R}$, defining the neighborhood between points, and a minimum number of neighbor points $np_{min} \in \mathbb{N}$, located in ϵ_{np}. The ϵ_{np} is set to the maximum inadmissible passage width between neighboring thunderstorms margins (e.g. 25 km) and can be determined by any distance function. Here, the Euclidean distance is used for calculation, resulting in a circular area around each point. The minimum density for identifying a cluster can thus be understood as the minimum number of neighboring points np_{min} per circular area $\epsilon_{np}^2\pi$. In a first step, the algorithm classifies all points as core, edge or noise points. A core point must have at least np_{min} within a radius of ϵ_{np} around it. Boundary points have fewer points than np_{min} in their neighborhood, but at least one core point. Since in the present case the surface density of the points is constant due to the homogeneous grid, $np_{min} = 1$ can be set. This implies that no boundary points exist. Noise points have fewer points than np_{min} in their neighborhood and are not adjacent to a core point. In the present case, there are no noise points, as the data is already filtered. After all points are classified, the densely connected points are determined from any core point, using a depth-first-search and are subsequently stored as nodes in a graph. Each individual graph represents a cluster. The algorithm runs until all core points are assigned to a cluster. DBSCAN has a worst case time complexity of $O(n^2)$, where n is the number of points.

Figure 3.5 depicts clusters of margins using $np_{min} = 1$ and $\epsilon_{np} = 25$ km generated on an equidistant grid spaced by 2000 m. Red and orange polygons are critical, as they represent

unpredicted thunderstorms. If they are outside the margins, they are not considered for trajectory planning. After the clusters are identified, concave polygons are generated.

3.1.5 Convex versus Concave Polygons

In many cases, merging (Section 3.1.3) or clustering of the margins (Section 3.1.4) around thunderstorms results in concave polygons. If the planner can only deal with convex polygons, this can result in negative consequences, regarding excessive mission area coverage. If concave polygons are converted to convex polygons, overlapping may occur occasionally whereby new concave polygons are formed. If this process is repeated until all polygons are convex, this potentially leads to a chain reaction, in which large mission areas are blocked. An example for this is shown in Figure 3.6. Merged combined margins from the example in Figure 3.5(a) are depicted in light blue. If they are all converted to a convex polygon, the covered area is increased by a factor of 1.25 (sum of all medium and light blue areas in Figure 3.6). If intersecting polygons are merged and transformed into convex form anew, the covered area is finally 1.63 times larger than initially (sum of all colored areas in Figure 3.6). Concave areas of thunderstorms should be avoided, as they frequently indicate dangerous phenomena (see Section 3.1.1). However, concave areas resulting from merged or clustered margins are classified as uncritical [83]. Therefore, they are not converted to convex polygons, as otherwise excessive coverage of free space is the consequence [83].

3.1.6 Spatio-Temporal Interpolation

A processed nowcast results in thirteen discrete sets of buffered thunderstorms. However, the MPTP requires discrete information concerning the obstacles for the intermediate query times spaced by Δt. This is achieved by applying a linear spatio-temporal interpolation. For instance in the upcoming Figure 6.2, the time increment is $\Delta t = 100$ s, which is why $T_N/\Delta t = 3600$ s/100 s= 36 level sets exist. Therefore, between each of the thirteen sets, two sets of interpolated polygonal shapes for each buffered thunderstorm have to be computed. An example for this procedure can be found in [83].

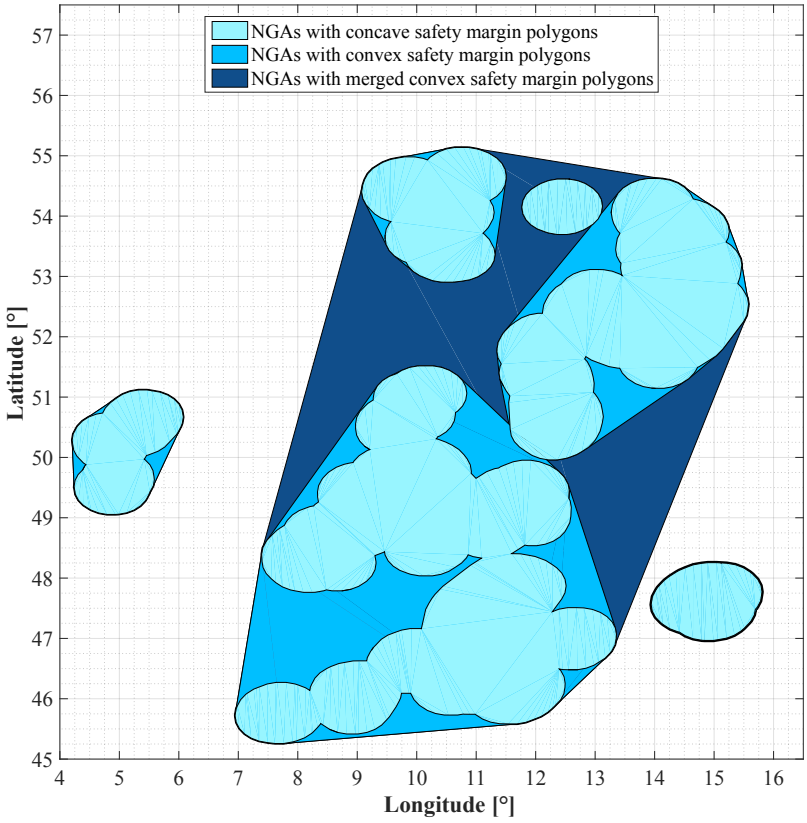


Figure 3.6: *Recursive transformation of concave to convex polygons can result in excessive coverage of free airspace, due to a chain reaction [83].*

3.2 Estimated Future Aircraft States

Future states of the aircraft are estimated in the earth fixed frame with the basic idea of a wavefront propagating with constant true airspeed V_T. The radial distance between isochronous state sets is $V_T \Delta t$ in the absence of wind. For the estimation of future states two cases have to be distinguished, namely wind speed $W = 0$ m/s or $W \neq 0$ m/s. In the real world, wind speed is rarely zero. What matters is, if the aircraft can compensate the wind.

If wind can be compensated future states primitives for up to one hour of flight can be preprocessed for different ground speeds V_G with $V_G = V_T = V$, for $W = 0$ m/s. For this purpose a 3-DoF simulation model of the aircraft is used. The aircraft's equation of motion in a flight path fixed coordinate system are given by

$$\dot{x} = V \cos\chi \cos\gamma, \tag{3.1a}$$

$$\dot{y} = V \sin\chi \cos\gamma, \tag{3.1b}$$

$$\dot{z} = -V \sin\gamma, \tag{3.1c}$$

with χ being the course or track angle and γ the angle of climb, integrated from

$$\dot{V} = g(n_x - \sin\gamma), \tag{3.2a}$$

$$\dot{\chi} = g\frac{n_z \sin\mu}{V\cos\gamma}, \tag{3.2b}$$

$$\dot{\gamma} = g\frac{n_z\cos\mu - \cos\gamma}{V}, \tag{3.2c}$$

where μ is the bank angle, n_x is the horizontal load factor and n_z is the vertical load factor, defined as

$$n_x = \frac{T-D}{mg}, \tag{3.3a}$$

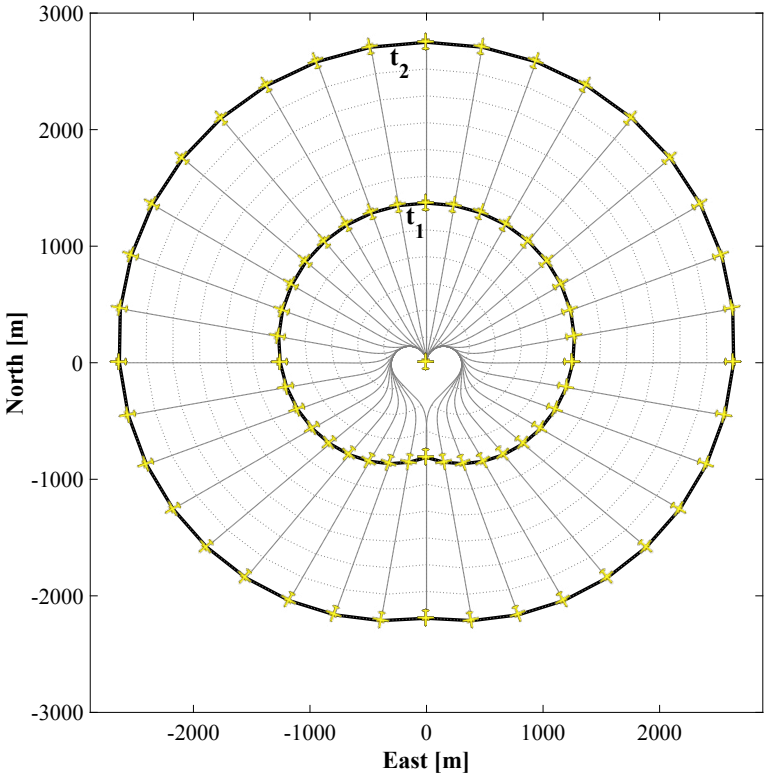
$$n_z = \frac{L}{mg}, \tag{3.3b}$$

with T being the thrust, D the drag, m the mass, g the earth's gravitation and L the lift.

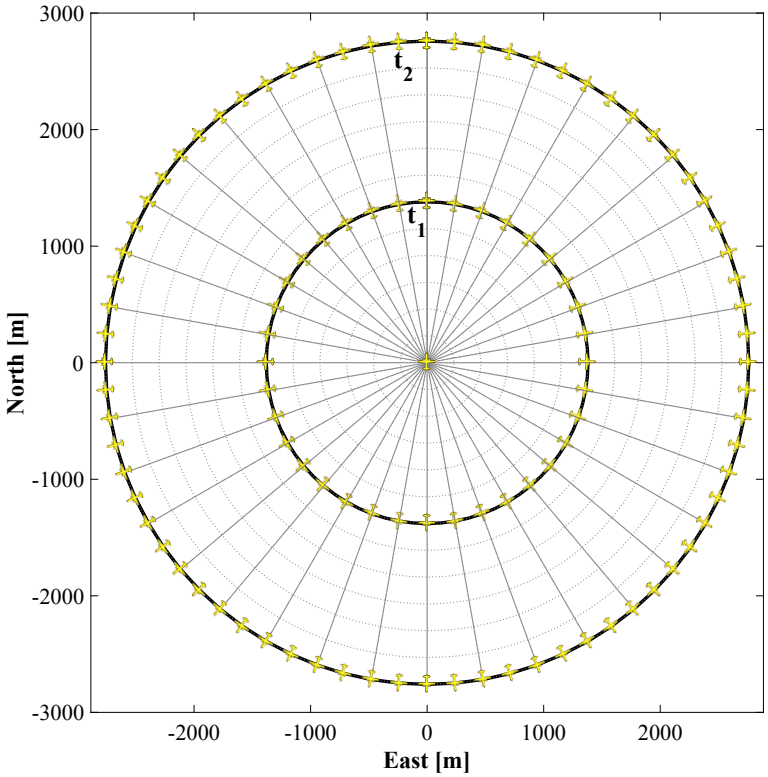
A controller, which consists of three laws for altitude, speed and azimuth control, keeps the aircraft model on commanded courses χ_{cmd}. These range from $\chi_{cmd} = 0°$ to $\chi_{cmd} = 360° - \Delta\chi$, linearly spaced by a course value of $\Delta\chi$. Starting from an initial state, in the origin of a Cartesian coordinate system heading north at time t_0, simulated future aircraft state samples are stored in time intervals of Δt. The resulting set of future aircraft state samples is denoted by A_{smp}. To obtain the set of future estimated aircraft states $A_{fut}(\mathrm{x}_{in})$ from an arbitrary initial state x_{in}, the A_{smp} are rotated around the z-axis according to the initial course χ_{in} and translated in all three spatial dimensions by a homogeneous transformation matrix H with $A_{fut}(\mathrm{x}_{in}) = H(\mathrm{x}_{in}) \cdot A_{smp}$

$$\begin{bmatrix} x_{fut} \\ y_{fut} \\ z_{fut} \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\chi_{in} & \sin\chi_{in} & 0 & x_{in} \\ -\sin\chi_{in} & \cos\chi_{in} & 0 & y_{in} \\ 0 & 0 & 1 & z_{in} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{smp} \\ y_{smp} \\ z_{smp} \\ 1 \end{bmatrix}. \tag{3.4}$$

The connection between contemporary aircraft state samples results in isochronous curves (black lines in Figure 3.7). The heart shape in Figure 3.7(a) results from turning-flight, when changing the initial course $\chi_{in} = 0°$ from north by $\pm 180°$. If the curve, due to the initial turning-flight when changing the course, is ignored, future aircraft states can be approximated with a marching wave expanding from an initial position p_{in} (see Figure 3.7(b)). The easiest way is to set up a matrix with linearly spaced values for each spatial dimension, which is applied in Chapter 6. The first dimension is the time



(**a**) *Exact future state samples.*



(**b**) *Approximate future state samples.*

Figure 3.7: *Future states are depicted by yellow aircraft symbols. The isochronous progress is shown with black lines for* $t_1 = 60$ *s and* $t_2 = 120$ *s. (a) Results of the flight simulation with constant velocity* $V_T = V_G = 23$ *m/s and a maximum bank angle of* $|\mu_{max}| = 19°$ *compared to (b) approximated states. In order to avoid a spatially decreasing density of state samples, as in (a), the number of sample states on the second ring in (b) is linearly increased by the factor two.*

t and the second one the course χ. However, with this method the resolution of sample states decreases with each isochronous ring. To obtain a more homogeneous distribution of the sample states, their number can be linearly increased, so that the k^{th} ring has k times the number of samples as the first. Hence, in the example in Figure 3.7(b), the number of states on the second isochronous ring is doubled. For the wind-free case the first-order nonlinear partial differential Eikonal equation can also be applied to compute the progress of the aircraft. It can be efficiently solved using the isotropic fast marching method (FMM) in $O(n \log n)$, with n being the number of grid points [84, 62]. This finite-difference method is suitable for online application. Using the multistencils second order fast marching method (MSFMM) improves the accuracy considerably [50].

However, for slow aircraft wind has a significant impact on trajectory planning, as it cannot always be completely compensated. Under the influence of wind, a constant V_T results in a spatially inhomogeneous V_G profile. Therefore, exact future states cannot be preprocessed as the resulting V_G depends on the position and course of the aircraft. In order to compute a trajectory, future states have to be computed several times (see Chapter 4). Due to the considerable computation time, the iterative simulation of exact future states is unsuitable for an online application. In case that a constant wind field is considered, approximate future aircraft states can be quickly computed, using a similar concept as in Figure 3.7(b). For a vectorized computation, a column vector containing all the course angles is set up. Additionally, three column vectors of the same size are created, of which one contains copies of the true airspeed, the other copies of the wind direction and the last copies of the wind speed. Then, the resulting ground speeds, when remaining on the respective courses, are computed, for example by passing the four vectors to Matlab's `driftcorr` function. As the wind is constant, the distance traveled on each radial is also constant. Thus, the ground speed column vector is copied in the time dimension and subsequently multiplied by a time matrix of the same size, whose rows are spaced by Δt. The resulting matrix contains the approximated aircraft progression in the constant wind field, for which Figure 3.8 shows an example. If the initial position p_{in} is not in the origin of the coordinate system, the future states are shifted by addition of its coordinates. As this method is fast, it is possible to recalculate future states based on the latest wind information, which is ideal for the present planning purpose.

If the wind is variable, instead, a noniterative ordered upwind method (OUM) can be applied for a fast estimation of future aircraft states [85, 86]. The implementation of this method is planned for a future version of the MPTP.

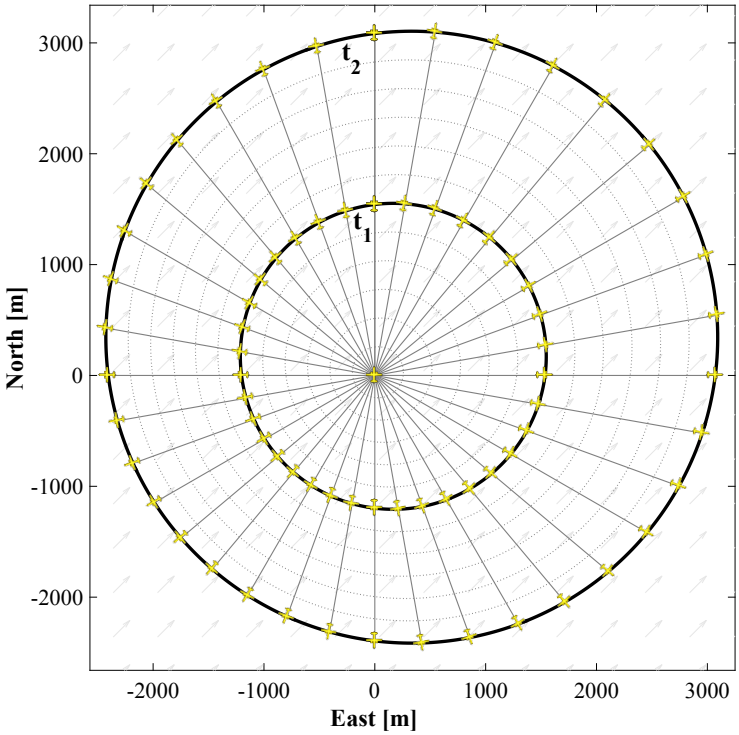


Figure 3.8: *Future aircraft states, when considering a constant wind field. The light gray arrows indicate the direction of the wind, which comes from* 225° *at* 4 *m/s. The true airspeed of the aircraft is* $V_T = 23$ *m/s and the time increment between the black isochronous progress is* $\Delta t = 60$ *s. In general it is sensible to use the prevailing mean wind direction and speed in the mission area.*

3.3 Estimated Future State Space

This section features Contribution 1: *Estimated Future Conflict Areas and Estimated State Space.*

Obstacle regions with invalid x_{fut} are denoted by X_{obs} [10]. Originating from an initial state $\mathrm{x}_{in} \in X$, where x_{in} is $(x_{in}, y_{in}, z_{in}, \chi_{in}, t_{in})$, deterministic samples of estimated future aircraft states are used to perform a conflict check with contemporaneous obstacles, to create an explicit representation of the estimated future obstacle region EX_{obs} (see Figure 3.11 and 3.12), whose definition is given below. This is done by superposition of the aforementioned prediction models for expected future obstacle location and estimated future aircraft states. Figure 3.9 depicts a set of simultaneous future aircraft states and predicted thunderstorms on different levels of time (vertical axis). The set of estimated conflicts from an initial state x_{in} is defined as the intersection of the set A_{fut}, consisting of future aircraft states x_{fut}, and set O, consisting of time-varying obstacles with arbitrary shape $O_n(t), n \in \{1, \ldots, m\}$.

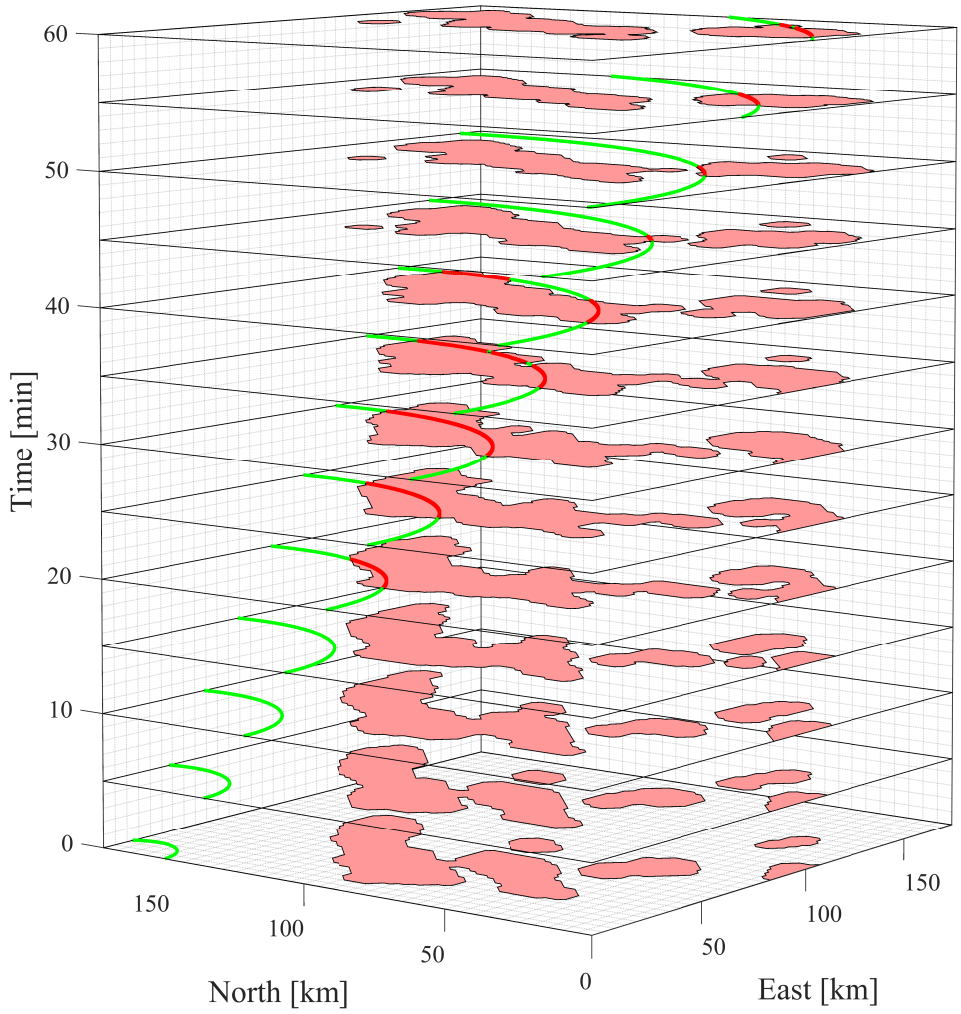


Figure 3.9: *Future aircraft states, which are not in conflict with obstacles (thunderstorms) are depicted in green, while conflicts are marked in red.*

Future states on and inside of time-varying obstacles

$$A_{fut}(\mathrm{x}_{in}) \cap O_n(t) \tag{3.5}$$

are enclosed by concave polygons, which are called estimated conflict areas (ECA), with $ECA_i(\mathrm{x}_{in})$ and $i \in \{1, \ldots, k\}$ being the the index of conflict areas. The estimated obstacle region, from an initial state, is the union set

$$EX_{obs}(\mathrm{x}_{in}) = \cup_{i=1}^{k} ECA_i(\mathrm{x}_{in}). \tag{3.6}$$

Alternatively to the deterministic samples, the ECAs can also be generated by computing the intersections between the isochronous curves (black lines in Figure 3.7) and the respective contemporaneous obstacles. A naïve intersection check takes $O(n^2)$, where n is the number of edges. If the number of actual intersections k is expected to be smaller than

n^2, it is faster to use a sweep line algorithm, which takes $O((n+k)\log n)$ operations [87]. However, if the number of k is of order $O(n^2)$, a sweep line algorithm takes $O(n^2 \log n)$ operations, which is worse than the naïve algorithm. The advantage of the intersection method is, that the representation of the exterior shape of the ECA is more accurate and less resolution dependent. However, as each polygon is represented by considerably less points, the shape of the polygons is also less unambiguous and thus not suitable for the creation of concave polygons. Besides the more robust ECA representation through state sampling, the in-polygon test, for which efficient implementations exist [88], runs in linear time [89]. Therefore, Matlab's `inpolygon` function is applied, which detects if points are located inside or on the edges of a polygon.

It is possible to limit the area, in which the aircraft is allowed to fly. The prescribed mission area for the motion planning is the workspace W. Then, the free estimated state-space is the difference

$$EX_{free}(\mathrm{x}_{in}) = W \setminus EX_{obs}(\mathrm{x}_{in}). \tag{3.7}$$

This operation has to be repeated for each iteration of the MPTP. If either x_s or x_g is inside the W but outside of $EX_{free}(\mathrm{x}_{in})$, it is covered by a thunderstorm.

Figure 3.11 shows an example for the evolution of EX_{obs} under the assumption of constant ground speed. The isochronous circles can be interpreted as level sets of a cone that opens perpendicular in the third dimension, that represents the time with the initial state x_{in} in the center. Contemporary conflicts are depicted in red. If the wind forecast (Section 3.1) is considered the isochronous progress is not circular anymore (see Figure 3.12). The shape of the obstacle region EX_{obs} is clearly different than in Figure 3.11.

The presented method can also be applied to determine EX_{obs} in non-level flight (Figure 3.10). For this purpose, three-dimensional obstacles are generated by extrusion of different 2D thunderstorm data i.e. ground and satellite based nowcast. Sufficiently large safety margins are added in all spatial directions. Where estimated future aircraft states are in conflict with an obstacle, an estimated conflict surface (ECS) is generated. For constant angle of climb γ, an approximate avoidance trajectory around moving three-dimensional obstacles can be computed for nonlevel flight, using the methods described in Chapter 4.

The previously introduced concept of estimated conflict areas is less safe, than those by [45, 90], where additionally inevitable collision states are considered. Therefore, the

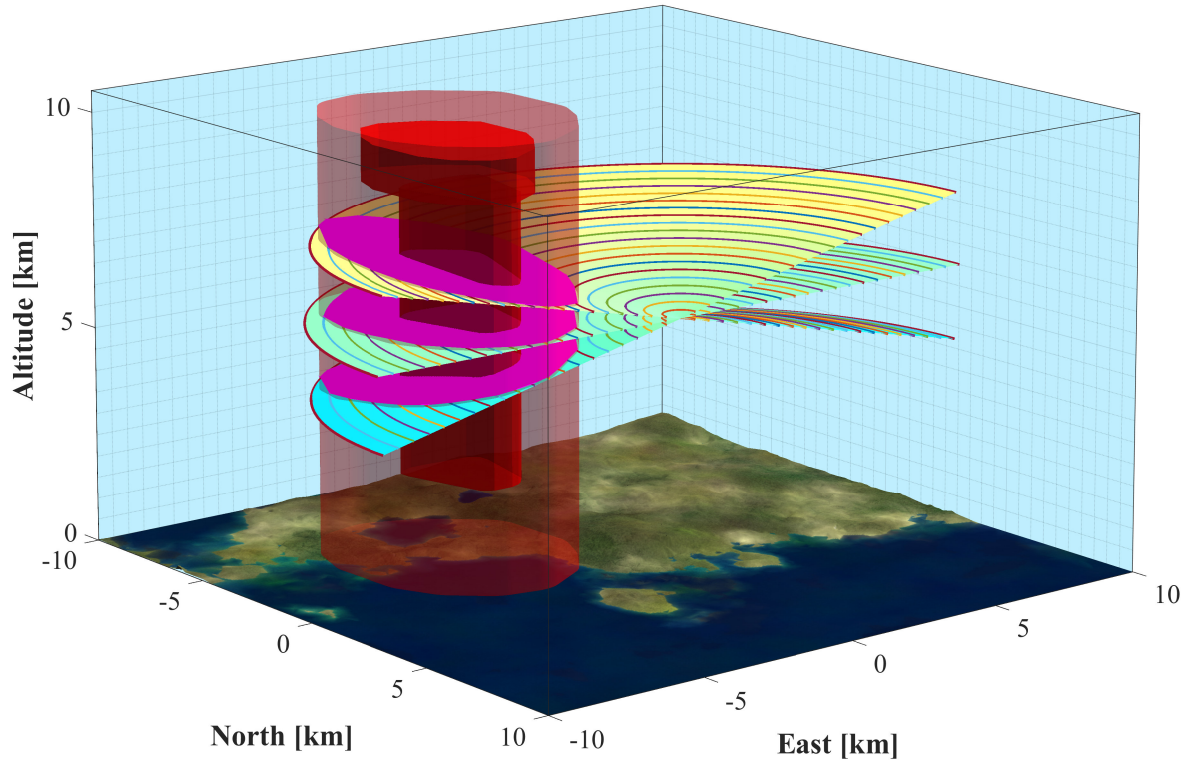


Figure 3.10: *The estimated conflict surfaces (magenta) for a 3D thunderstorm cloud (red volume) and its margin (translucent red volume) are shown for climb, level flight and descent. In this case the set of future aircraft states A_{fut} and future obstacles O are both $\subseteq \mathbb{R}^4$ (a tupel of (x, y, z, t)), while the workspace W and estimated obstacle region EX_{obs} are both $\subseteq \mathbb{R}^3$ (a tupel of (x, y, z)).*

only areas where safety amidst ECAs can be guaranteed, are sectors between tangent lines from the initial state to different ECAs. These are called estimated safe areas ESAs (transparent green areas in Figure 3.11). In case, that no trajectory plan can be computed, in an allotted time interval (see Chapter 2), it is safe to proceed straight flight, i.e. normal to the projected isochronous rings, inside an ESA. This is a straightforward form of partial planning [31], which procures time for replanning, without endangering the aircraft. In combination with the ability to plan holding patterns (see Section 5.4), this provides the aircraft's ability to act.

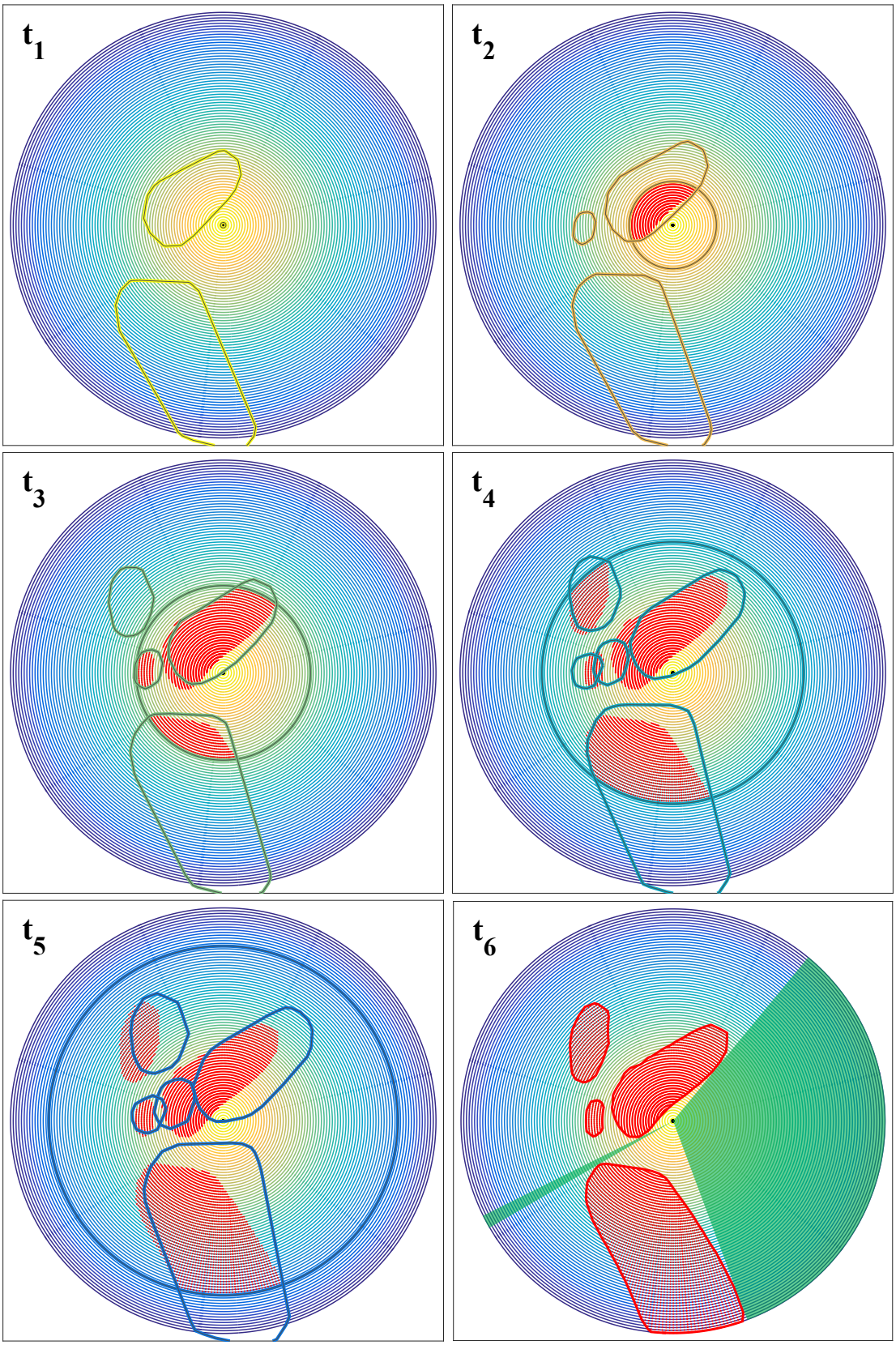


Figure 3.11: *In this example the ground speed is equal to true airspeed. The evolution of estimated conflict areas is illustrated at six points in time* $(t_1, \ldots, t_6)$*. For each time the estimated future states (highlighted isochrone circle) in conflict with contemporaneous expected thunderstorms (varying polygon shapes) are marked in red. This results in the final shape of the ECAs, shown at* t_6*, where each estimated safe area ESA is depicted in transparent green.*

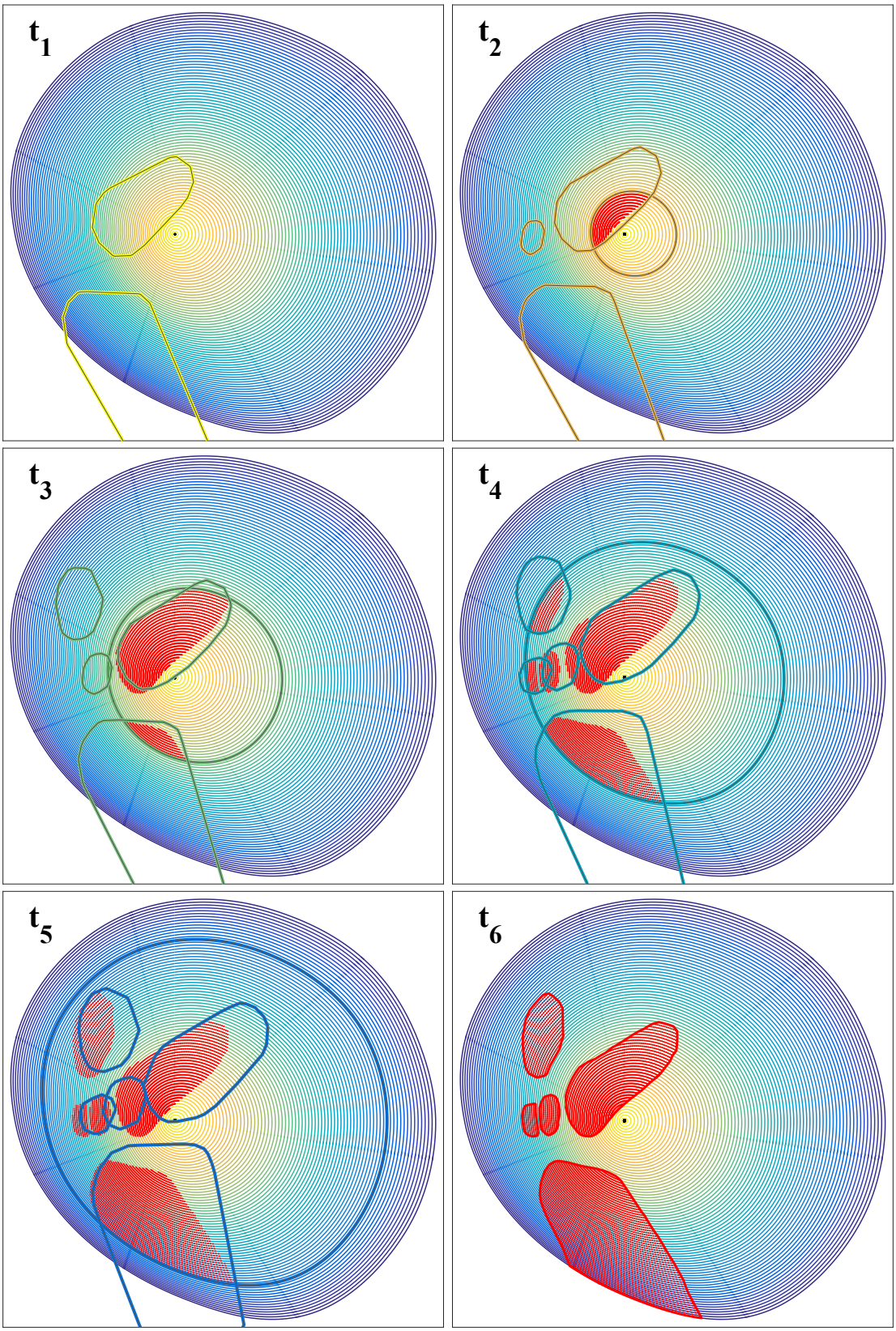


Figure 3.12: *In the anisotropic case, the groundspeed is not equal to true airspeed and depends on the aircraft's position and course. The inhomogeneous gridded wind data is taken from a GRIB file (see Section 3.1). When considering wind, the resulting ECAs differ noticeably from those in Figure 3.11.*

Chapter 4

Optimization - Trajectory Computation

This chapter introduces the necessary steps to compute a time-monotone, resolution-optimal trajectory through predicted, time-varying thunderstorms, under the consideration of nonholonomic turning-flight constraints. A combinatorial planning method is introduced, for which Section 4.1 describes the discrete representation of the search space and Sections 4.2 and 4.3 the searching procedure, to effectively find an optimal solution.

4.1 Graph of Free Estimated State Space

A discrete roadmap of the estimated state space $EX_{free}(\mathrm{x}_{in})$, which in the presence of obstacles is basically a polygon with holes (see Section 3.3), is created in form of a visibility graph $VG(\mathrm{x}_{in}) = (V, E)$ [10]. It is only constructed, if the direct connection between the initial x_{in} and the goal state x_g intersects an obstacle. A visibility graph contains vertices $v \in V$ and edges $e \in E$, which are undirected and weighted.

Generally in a static avoidance problem in C-space, the search space can be described by a single VG. However, in the present dynamic problem in X-space, the search space is usually build gradually from individual VGs, which is described in detail in Section 4.2.2. However, in both static and dynamic application the search space representation via VG has the same advantages. A complete VG contains the optimal solution, if one exists. Its construction is deterministic and therefore results are reproducible. Additionally visibility based paths or trajectories are intuitive to humans. The application of a VG is most suitable for two-dimensional problems with a polygonal obstacle representation, which corresponds to the problem at hand. A disadvantage of VGs is, that trajectories can

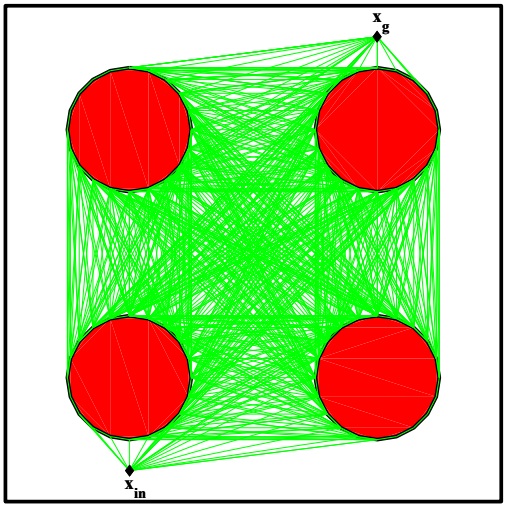
pass directly alongside obstacles, which for the present application poses no problem, as the actual obstacles are thunderstorms, surrounded by safety margins (see Section 3.1). Another disadvantage is the computational load associated with its construction. The lower bound for the computation of a complete VG is $\Omega(|V|^2)$ or $\Omega(|E|)$ in big Omega notation (see Chapter 1.3), where $|V|$ is the total number of obstacle vertices and $|E|$ is the resulting number of edges [57]. However, sophisticated algorithms with these bounds, can be difficult to implement. The time complexity of a more simple algorithm, like the rotational plane sweep algorithm, is $O(|V|^2 \log |V|)$ [55].

In this section several methods to reduce the computational effort are presented. Among others, this is the use of a so-called reduced visibility graph (VG_r), which exclusively contains tangent/bitangent edges (see Section 4.1.2) [37]. Ideally, a VG only consists of tangent edges, between points and obstacles, bitangent (supporting and separating) edges, between obstacles and bitangent edges on obstacles. Methods for the construction of partial or complete reduced visibility graphs are introduced, which work for both convex and nonconvex polygons. Figure 4.1 shows a comparison between three basic visibility graph variants, inside a mission area (black square) and outside of obstacles (four red icosagons). The unreduced visibility graph $VG(\mathrm{x}_{in})$ contains all visible edges. A VG_r contains exclusively tangent and bitangent visible edges, while a partial visibility graph $VG_p(\mathrm{x}_{in})$ only connects the initial state x_{in} to those vertices $v \in V$, which are connected by a visible tangent edge. The large difference in visible edges (green lines) in this simple example, is the reason, why only the methods in (b) and (c) are considered for trajectory planning.

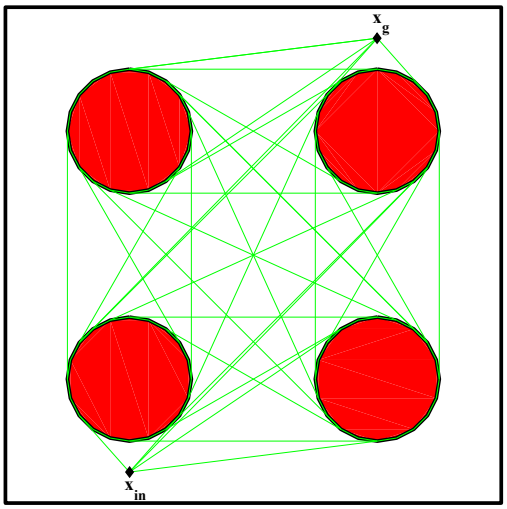
As mentioned before, the time complexity for the construction of a visibility graph can be expressed by its number of vertices $|V|$ and edges $|E|$. So can the time complexity of a graph-search algorithm, which is suitable for searching the VG. The number of vertices $|V|$ and edges $|E|$ are directly related. Fewer $|V|$ result in fewer $|E|$ and vice versa. As several visibility operations have to be constructed and searched, in order to compute a trajectory (see Section 4.2), their efficient computation is crucial for the performance of the trajectory planner. In the following Sections 4.1.1 and 4.1.2, different methods for this purpose are presented.

4.1.1 Reduction of Vertices on Obstacles

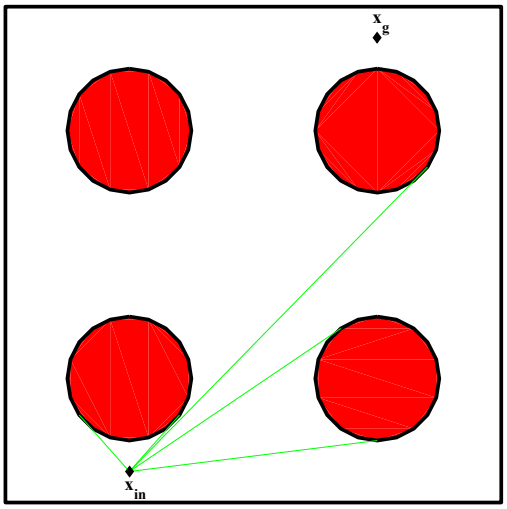
Polygonal obstacles (nowcasted thunderstorms) are frequently jagged (see Figure 4.2(a)), which means, that they contain a large number of dispensable vertices. By default,



(**a**) *Unreduced visibility graph VG.*



(**b**) *Reduced visibility graph VG_r.*



(**c**) *Partial visibility graph VG_p.*

Figure 4.1: *(a) The unreduced VG contains all visible edges, of which a large number is unnecessary in order to find the shortest connection from x_{in} to x_g. (b) The VG_r exclusively contains tangent edges, between states (black diamonds) and obstacles, as well as bitangent edges, between and on obstacles. (c) The partial VG_p only contains the visible tangent edges of first order, between initial state x_{in} and obstacles.*

predicted thunderstorms are enlarged by probabilistic and/or safety margins (see Section 3.1) to model the considerable amount of uncertainty in the nowcast (see example in Figure 6.1). Therefore, these margins do not represent exact boundaries. Thus, a certain degree of geometric inaccuracy, due to line smoothing, is acceptable. In order to reduce the number of vertices $|V|$ in the estimated state space, the Douglas-Peucker-algorithm can be applied to obstacles, before a VG is generated [91], which automatically reduces $|V|$ in the VG. It should be noted that thereby a complete planning algorithm becomes resolution-complete. This means, that it is still guaranteed to find a solution if one exists, however at the lower level of resolution [37, 10]. The line-smoothing algorithm uses a tolerance band $\epsilon > 0$, orthogonal to the left and right of an examined curve, on a equirectangular map projection. Under the assumption that the earth is a sphere, ϵ is converted from spherical distance D_m in meters to D_d in degrees by

$$D_d = \left(\frac{D_m}{R_e}\right) \frac{180}{\pi} [^\circ], \tag{4.1}$$

where D_m is the great circle distance and $R_e = 6371000$ m is the mean earth radius. The algorithm examines the ordered vertices $V = (v_1, \ldots, v_n)$, of a curve. At the beginning, the vertex v_f, which is orthogonally furthest away from the line segment between the first and last vertex of the curve, is searched. If v_f lies within ϵ, all vertices between v_1 and v_f are removed and a line remains. If v_f lies outside of ϵ, the function is recursively called, until the whole curve has been examined. Vertices within ϵ are be removed from the list. The output of the algorithm is a subset of V. The worst-case time complexity of the original Douglas-Peucker algorithm is $O(nm)$, where n is the number of vertices of the input curve and m the number of vertices of the simplified curve. To avoid topological inconsistencies and to ensure homeomorphism between the original and reduced shape, the method introduced by [92] can be applied, which also runs in $O(nm)$. In [93] an algorithm is introduced, which bounds the worst-case running time to $O(n \log n)$, which is the expected case for the original algorithm. Topological problems can be avoided, by limiting the maximum value for the tolerance band. Through an automatic selection of an appropriate tolerance band ϵ, a reduction in $|V|$ can be achieved, under preservation of the shape, e.g. by an error bound termination condition [94].

Figure 4.2 shows an example for a reduction of vertices, by approximately 60 %, while the general shape preservation is good. The maximum lateral error in this example is $\epsilon = 0.022^\circ \approx 1.31$ NM. The error in the outline length of all obstacles is approximately 4 %. As mentioned before, predicted thunderstorms are enlarged by different discrete

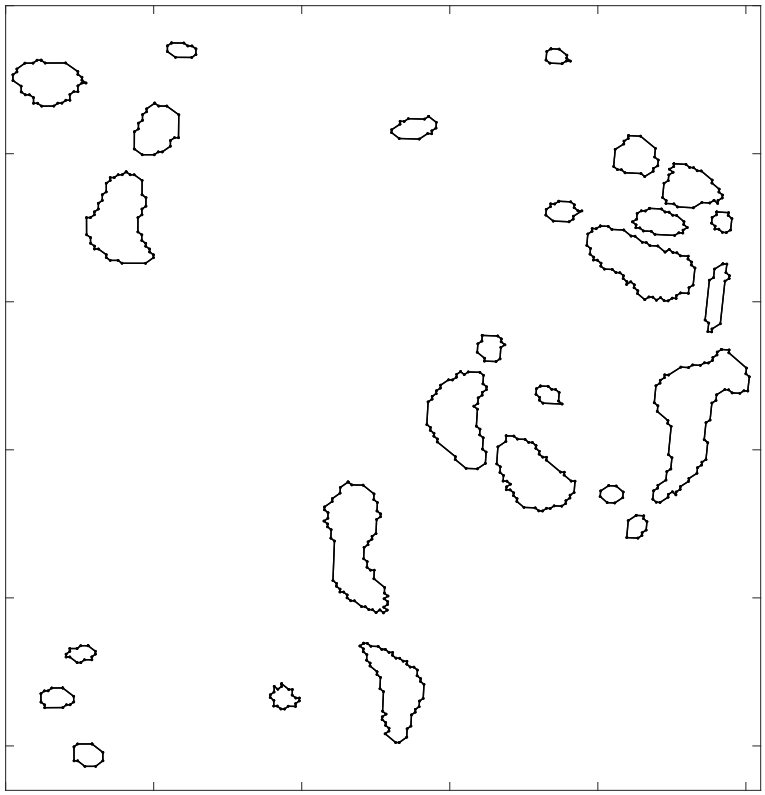
margins, to model the considerable amount of uncertainty in the nowcast (see Section 3.1). In worst case, the recommended margin of 20 NM [80], is undercut by the aforementioned lateral error. In order to ensure, that a prescribed margin is never undercut, the tolerance band ϵ can be added to the margin, as shown in Figure 4.3.

4.1.2 Reduction of Edges in a Visibility Graph

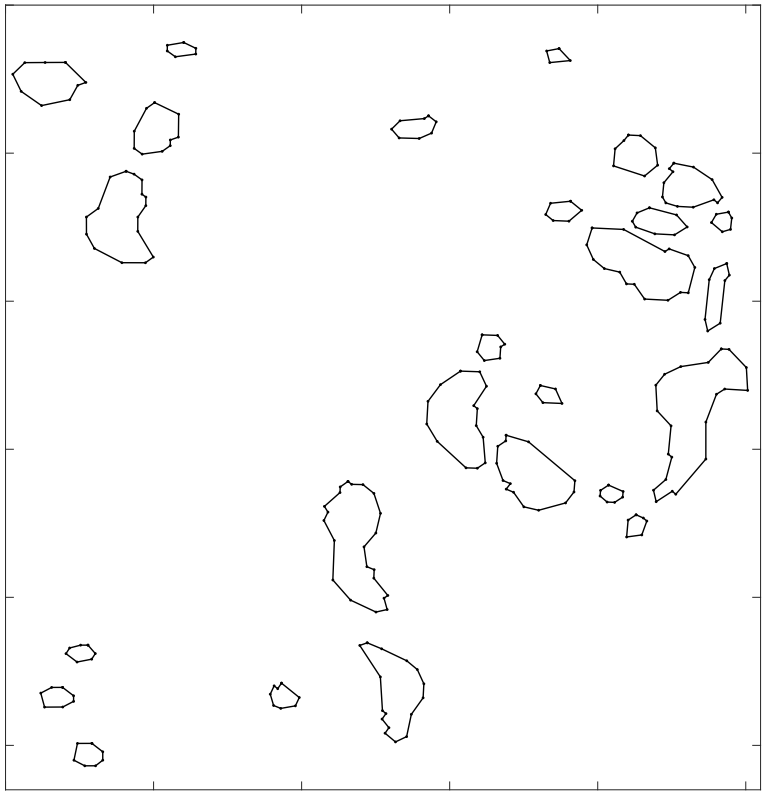
This section features Contribution 2: *Matrix Computation of Visible Tangent and Bitangent Edges* and Contribution 3: *Partial Shortest Path Map.*

A visibility graph, which exclusively contains tangent edges between points and obstacles, and bitangent edges between the obstacles, is called a VG_r [37]. A VG_r is complete in the sense that it contains all connections, which are necessary to find an optimal solution, e.g. the shortest path. Such a graph can be computed using a naïve approach in $O(|E|^3)$, where $|E|$ is the number of obstacle edges. Alternatively, a rotational sweep algorithm can be applied, which has a time complexity of $O(|E|^2 \log |E|)$ [95]. This algorithm can also be used to compute a VG_r [10]. In [54] a method is introduced for the special case of exclusively convex polygons and two arbitrary points, i.e. start and goal. It has a time complexity of $(|V| + |P|^2 \log |V|)$, where $|V|$ is the number of vertices and $|P|$ the number of disjoint polygons. In [96] the same time complexity for the construction of a VG_r in a polygon with holes is achieved for the general case with concave shapes. This is achieved by first computing the convex form the concave obstacles and then applying the algorithm in [54]. This algorithm is optimal, i.e. lower time complexity than $O(|V|^2)$, if the number of obstacles $|P|$ is relatively small. For the avoidance of thunderstorms, $|P|$ is generally significantly smaller than the number of vertices $|V|$. Therefore, this is the fastest algorithm for the problem at hand. However, the implementation is not straightforward. The optimal time complexity to compute an unreduced VG for general disjoint polygons, is $O(|E|^2)$ [97, 98]. Additionally, [98] introduced a method, for sparse visibility graphs, which runs in $O(|E| \log |V|)$, where $|E|$ is the number of edges in the resulting visibility graph. For the interested reader [99] is recommended, where several of the aforementioned algorithms are implemented and compared.

In the following, the number of edges of a visibility graph is reduced by preprocessing exclusively tangent and bitangent edges, using matrix operations. Tangent edges are determined in $O(|V_{EX}|)$ and bitangent edges in $O(|V_{EX}|^2)$, where V_{EX} are the unique vertices in the estimated state space EX. Consecutively a method is described, which determines the visibility of these edges solely by counting the number of intersections with



(**a**) *Original thunderstorm polygons, from a nowcast.*



(**b**) *Smoothed shapes, using Douglas-Peucker algorithm.*

Figure 4.2: *Line-smoothing applied to polygons of a thunderstorm nowcast. While all shapes are well preserved, the overall number of vertices is reduced from 704, in (a), to 285, in (b).*

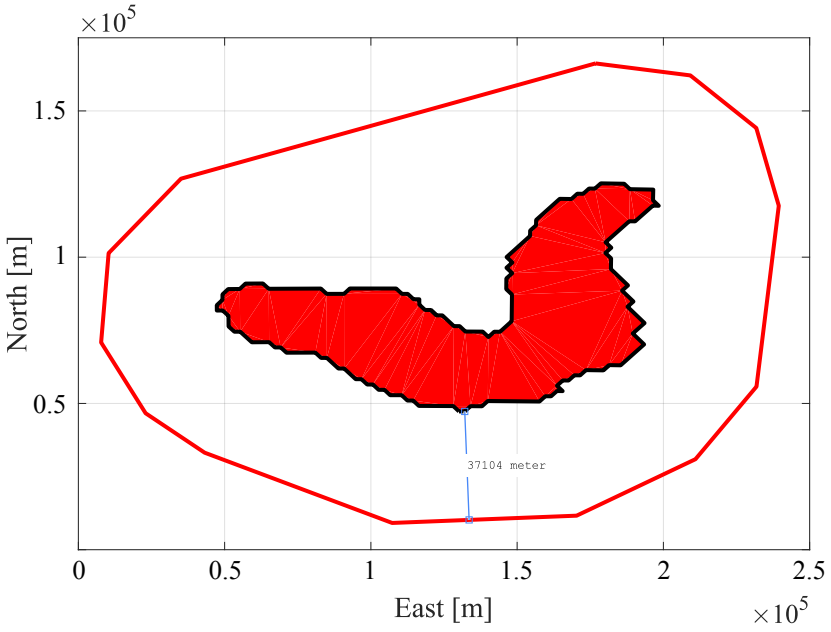


Figure 4.3: *In this example, a relatively large tolerance band of* $\epsilon = 0.044°$ *is added to the* 20 *NM (*37040 *m) lateral clearance, recommended by the Federal Aviation Administration [80]. Thus, the smoothed margin never falls below the* 20 *NM clearance (blue line measurement).*

obstacle edges, using an algorithm for the general red-blue intersection problem. The resulting visibility graph is a VG_r[37]. This procedure has two major advantages: under certain conditions, for example a high sparsity regarding the tangent and bitangent edge candidates, the generation of a VG_r can be efficient. Secondly, in contrast to a complete visibility graph VG, a reduced visibility graph VG_r contains a lot less unnecessary edges (although some bitangent edges lead nowhere, see Figure 4.4(a)), which reduces the mean branching factor, i.e. number of outgoing edges and therefore states to examine. This significantly reduces both the runtime and memory requirements of an ensuing graph search, using e.g. Dijkstra's algorithm or A*-search.

Selection of Tangent and Bitangent Edges

To determine tangent edges from start (initial) and goal state to the obstacles, a tangent edge angle matrix (M_{ta}), containing the angles of all possible tangent edge candidates measured in the canonical way, is set up. Additionally, two matrices of the same size are set up, which contain all the relative angles (to those in M_{ta}) of the corresponding adjacent edges on the obstacles. They are called previous edge angle matrix (M_{pa}) and next edge angle matrix (M_{na}). The matrix M_t contains the logical values, which indicate if an edge (subscript i, j) in M_{ta} is tangent (logic 1) or not (0). The number of unique

tangent edge candidates E_{tc} in M_t, outgoing from start and goal state to the vertices of the estimated state space V_{EX}, is given by

$$|E_{tc}| = 2|V_{EX}|. \tag{4.2}$$

The dimension of M_{ta}, M_{pa} and M_{na} is $2 \times |V_{EX}|$.

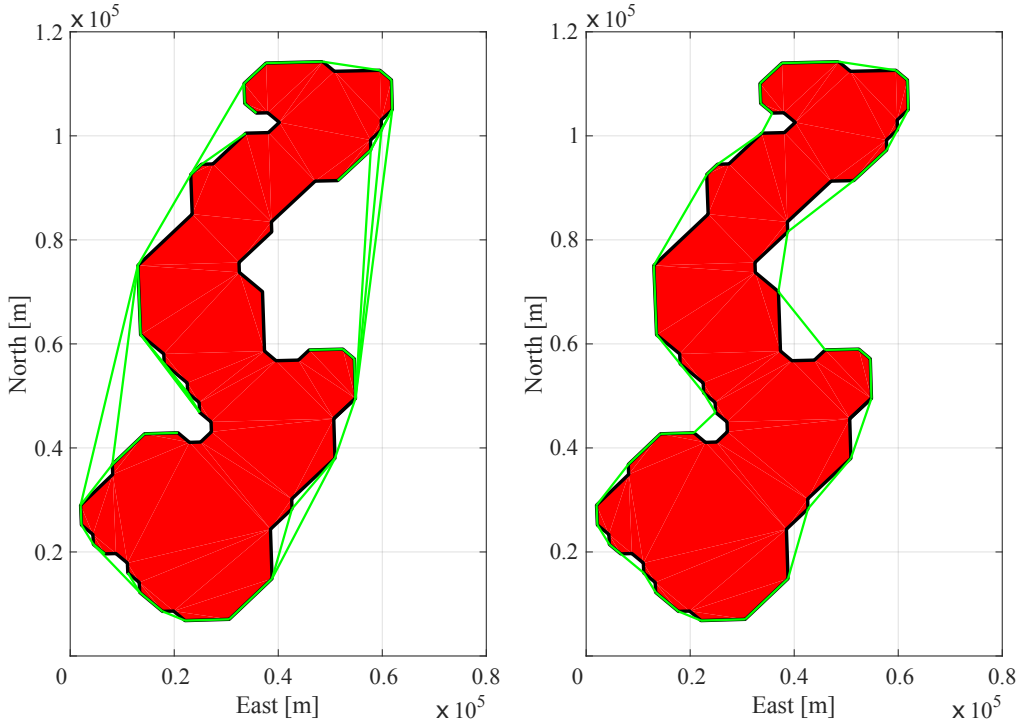
In order for an edge to be tangent, the trigonometric conditions are that the relative angles in $M_{pa_{i,j}}$ and $M_{na_{i,j}}$ are both either $((0 \leq M_{pa_{i,j}} < \pi) \wedge (0 \leq M_{na_{i,j}} < \pi)) \vee ((\pi \leq M_{pa_{i,j}} < 2\pi) \wedge (\pi \leq M_{na_{i,j}} < 2\pi))$. Figure 4.5 depicts the approach. Four edges, starting from p_7 to the vertices of polygon P_3, are examined. The relative angles to the adjacent edges of P_3 are determined in counterclockwise sense (black circle segments). In this example, the edges from p_7 to p_9 and p_7 to p_{10} fulfill the aforementioned conditions and thus are tangent.

For the computation of M_{ta}, M_{pa}, M_{na} and M_t each time $2|V_{EX}|$ operations are performed. Thus, the time complexity to determine the exclusively tangent edge candidates, between the start state and goal state to the obstacles, is $O(|V_{EX})|$.

The selection of bitangent edges follows next. A square matrix (M_{bta}), built from the vertices of estimated state space V_{EX}, contains the angles of all bitangent edge candidates. Analogously to the tangent edge selection, two additional matrices of the same size, as M_{bta}, are set up, which contain all the relative angles of the corresponding adjacent edges on the obstacles. They are again called previous edge angle matrix (M_{pa}) and next edge angle matrix (M_{na}). The matrix M_{bt} contains the logical values, which indicate if an edge in M_{bta} is bitangent or not. The number of unique bitangent edge candidates $|E_{btc}|$ corresponds to the size of the upper triangular matrix, above the diagonal of M_{bt} and is given by

$$|E_{btc}| = \frac{|V_{EX}|^2 - |V_{EX}|}{2}. \tag{4.3}$$

An exemplary M_{bt} is depicted on the right side of Figure 4.5. It is built from the vertices of the three polygons P_1, P_2 and P_3. The diagonal represents the vertices themselves. All possible undirected edge combinations exist. The entries of the matrix, which belong to the respective polygons on the left side of the figure, are colorized accordingly. Each entry of M_{bta} contains either a logical 0 or 1. This indicates whether an edge, starting from the row-index, encounters the necessary trigonometrical conditions to be tangent, at the column-index.



(**a**) *Bitangent edges, of a concave polygon.* (**b**) *Polygon, built by convex vertices.*

Figure 4.4: *(a) Exclusively bitangent edges on an obstacle after one iteration. Some dead ends and unnecessary edges (unnecessary, in order to find the shortest path or trajectory) may exist. The convex hull of an obstacle is always included. If this procedure is repeated iteratively, until the number of vertices does not decrease (in which case no more concave vertices exist), the convex shape of a concave polygon can be computed. (b) By filtering the concave vertices of the obstacles, a new polygon (closed shape by green lines) is created, which reduces the number of vertices in the estimated state space.*

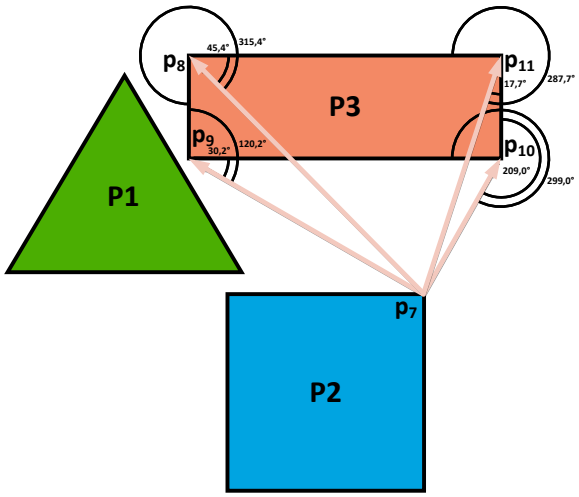
In order for an edge in M_{bta} to be bitangent, the same trigonometric conditions as before apply. However, this time they have to be true in the upper and lower triangular part of M_{bt}. Therefore, a test edge is bitangent, if both the entry and its symmetric counterpart are true (see green entries in Figure 4.6). The relative angles $M_{pa_{i,j}}$ and $M_{pa_{i,j}}$ have to satisfy

$$((0 \leq M_{pa_{i,j}} < \pi) \wedge (0 \leq M_{na_{i,j}} < \pi)) \vee ((\pi \leq M_{pa_{i,j}} < 2\pi) \wedge (\pi \leq M_{na_{i,j}} < 2\pi)) \quad (4.4)$$

and the relative angles $M_{pa_{j,i}}$ and $M_{na_{j,i}}$ also have to satisfy

$$((0 \leq M_{pa_{j,i}} < \pi) \wedge (0 \leq M_{na_{j,i}} < \pi)) \vee ((\pi \leq M_{pa_{j,i}} < 2\pi) \wedge (\pi \leq M_{na_{j,i}} < 2\pi)). \quad (4.5)$$

In the example in Figure 4.5, only the edge between p_7 and p_9, is bitangent, as both entries ($i = 9$, $j = 7$ and $i = 7$, $j = 9$) in the matrix on the right side are true.



	p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}	p_{11}
p_1	-	1	1	0	1	0	1	1	1	0	0
p_2	1	-	1	0	1	0	1	1	0	1	0
p_3	1	1	-	1	0	0	1	0	1	1	0
p_4	0	1	1	-	1	0	1	0	1	1	0
p_5	0	1	1	1	-	1	0	0	1	1	0
p_6	1	1	0	0	1	-	1	0	1	1	0
p_7	1	1	0	1	0	1	-	0	1	1	0
p_8	1	0	1	0	1	0	1	-	1	0	1
p_9	1	0	1	0	1	0	1	1	-	1	0
p_{10}	1	0	1	1	0	1	0	0	1	-	1
p_{11}	1	0	1	1	0	1	0	1	0	1	-

Figure 4.5: *The applied trigonometric condition scheme, to determine bitangent edges between polygons, is depicted on the left side. The angles between the test edges from vertex p_7 to the vertices of polygon P_3 and the respective adjacent edges to the aforementioned vertices are computed. The four test edges are marked by a black box, in the logical matrix on the right. Only the edge between vertex p_7 and p_9 is bitangent, as also p_9 to p_7 is true.*

Additional information is stored for each bitangent edge in order to deal with degeneracies, i.e. collinearity. If an bitangent edge is directly on an obstacle, it is marked as body edge. Also the number of adjacent collinear obstacle edges (c) is stored for each of bitangent edge. If either the angle to a previous edge is 0 or π, or the angle to a next edge is 0 or π, the respective entry is set to true. If either in the upper or lower triangular part of M_{pa} or M_{na} collinearity is detected, then $c = 1$. If in both the upper and lower triangular part of M_{pa} or M_{na} collinearity is detected, then $c = 2$. This information is required to determine the visibility of bitangent edges, exclusively by the number of intersections with obstacle edges, which is described later on.

For the computation of M_{bta}, M_{pa}, M_{na} and M_{bt}, each time $|V_{EX}|^2$ operations are performed. Thus, the time complexity to determine the exclusively bitangent edges between the obstacles is $O(|V_{EX}|^2)$.

If the algorithm to detect bitangent edges is iteratively applied to a concave shape, its convex hull can be computed stepwise. Depending on the number of concave polygons and their degree of concavity, up to half of the bitangent edges $|E_{bt}|$, between the obstacles in estimated state space EX, consist of at least one reflex (concave) vertex. In any case, such a bitangent edge intersects the obstacle, containing the reflex vertex. Therefore, these edges are directly omitted, as their only purpose is to slow down the al-

gorithm. By excluding reflex vertices, the number of bitangent edges E_{btc}, in the example in Figure 4.10(b), is reduced from 11648 to 5581, from which only 1273 are visible anyway. Furthermore, the exclusion of reflex vertices reduces the overall number of resulting vertices $|V_{VG}|$ in the visibility graph. However, one problem arises, when reflex vertices of the workspace W are filtered, as the outer limit of the estimated state space EX is concave. If an obstacle is merged with the workspace, as in the lower right side of the examples in Figure 4.10, this leads to incorrect results. However, by converting the order of vertices in the polygon, which contains the boundary of the workspace (mission area), from counterclockwise to clockwise sense, this issue is resolved.

Sparsity of the Tangent and Bitangent Matrices

A matrix is said to be sparse, if it contains more zero than nonzero elements (NZ). The sparsity S of a matrix M is calculated by

$$S(M) = \frac{E(M) - NZ(M)}{E(M)}, \tag{4.6}$$

where $E(M)$ is the total number of elements in the matrix M.

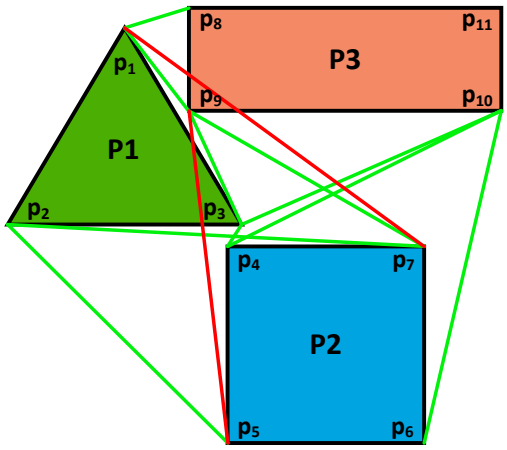
Table 4.1 shows exemplary values for the sparsity of M_t and M_{bt} in ten estimated state spaces EX, similar to the one depicted in Figure 4.10. Especially the sparsity M_{bt} is usually high, with $S(M_{bt}) \geq 90$ %. This property is advantageous, as only the tangent and bitangent edges have to be consecutively checked for intersection against the edges of obstacles.

Table 4.1: *Sparsity of the tangent and bitangent edge candidate matrices, using nowcast data from the 2015/07/07. Neither line-smoothing nor a limitation of the workspace is applied. The mean sparsity for the tangent matrices is $S(M_t) = 83.42$ % and for the bitangent matrix is $S(M_{bt}) = 96.02$ %.*

Time [h]	**Sparsity of M_t** [%]	**Sparsity of M_{bt}** [%]
12:00	79.29	94.75
13:00	81.92	95.30
14:00	80.97	95.55
15:00	84.78	95.77
16:00	84.10	96.09
17:00	84.99	96.24
18:00	85.22	96.65
19:00	83.38	96.74
20:00	84.36	96.10
21:00	85.27	96.45
22:00	83.32	96.62

Visibility Determination for Preselected Edges

After the tangent and bitangent edges are preselected, their visibility is subsequently determined. An edge is visible, if it is only intersected at its endpoints, exclusively by the body edges of obstacles, which share an endpoint. In the example in Figure 4.6, two bitangent edges are invisible due to intersections (red lines on the left side and red entries on the right side). The naïve approach is to perform intersection checks for each



	p1	p2	p3	p4	p5	p6	p7	p8	p9	p10	p11
p1	-	1	1	0	1	0	1	1	1	0	0
p2	1	-	1	0	1	0	1	1	0	1	0
p3	1	1	-	1	0	0	1	0	1	1	0
p4	0	1	1	-	1	0	1	0	1	1	0
p5	0	1	1	1	-	1	0	0	1	1	0
p6	1	1	0	0	1	-	1	0	1	1	0
p7	1	1	0	1	0	1	-	0	1	1	0
p8	1	0	1	0	1	0	1	-	1	0	1
p9	1	0	1	0	1	0	1	1	-	1	0
p10	1	0	1	1	0	1	0	0	1	-	1
p11	1	0	1	1	0	1	0	1	0	1	-

Figure 4.6: *Only edges, which are symmetrically tangent (logical 1), are bitangent (green entries in the right M_{bt}). Two bitangent edges are eliminated (red lines, on the left), due to intersections with polygons P_1 and P_3. The remaining bitangent edges are visible (green lines).*

preselected edge against all obstacle edges E_{EX} in the estimated state space, which results in a time complexity of $O((|E_t| + |E_{bt}|)|E_{EX}|)$.

In the following a novel method is introduced to determine the visibility of the preselected E_t and E_{bt} solely by counting the number of intersections by obstacle edges E_{EX}, using the information stored in the previously presented matrices. The prerequisite is an algorithm that counts the number of intersections between to sets of edges. Suitable for this purpose is the algorithm by [100], which is able to count the number of intersections between a set of red edges E_R and a set of blue edges E_B in $\mathbb{R}^2$, for the general case, in which monochromatic intersections are admissible. Here, the tangent and bitangent edges form the red set E_R, while the edges of the obstacles form the blue set E_B. Counting the number of intersections of all edges in E_B (obstacle edges) with each edge in E_R (tangent and bitangent edge) take approximately $O(|E|^{4/3} \log |E|)$, where $|E| = |E_t| + |E_{bt}| + |E_{EX}|$. As the intersection counting with the red-blue method is expensive, it is not suitable for

large scenarios, with dense tangent and bitangent matrices. Table 4.1 indicates that a high sparsity is present in the problems at hand, which is why such an approach is suitable for the computation of reduced visibility graphs VG_r. Since the implementation of the aforementioned algorithm is not straightforward, Matlab's `polyxpoly` function is applied. The input arguments are the two sets of edges E_R and E_B, which are checked for intersection. By using the output argument vector, which contains the related indices of the edges leading to an intersection point, it is possible to count the number of intersections. In order to obtain correct results, it is important not to use the `unique` option. A feature of `polyxpoly` is that it reports intersections of collinear edges. However, in the following a solution is presented for the general case that intersections of collinear edges are not detected. It is demonstrated, that the visibility of tangent and bitangent edges can be determined, solely by the number of obstacle edges they are intersected by, and that this results in a complete VG_r.

A tangent edge always ends on a vertex of an obstacle. Therefore, the maximum number of intersections for a visible tangent edge is $k_t = 2$. If a tangent edge is collinear with an obstacle edge, then $k_t = 1$. In case, that the edge is additionally intersected by a further obstacle, the minimum number of intersections is $k_t = 3$. By setting the condition $k_t \leq 2$, the visibility can be clearly determined.

A special case are bitangent edges, whose two vertices are sequential on the same polygonal obstacle. They are called body edges, as they are located directly on an obstacle. To determine their visibility three cases have to be distinguished. This allows to deal directly with degeneracies without using perturbation techniques, as recommended by [101]. The maximum number of intersections of a body edge is two. The number of required intersections for visibility is $k_{bt} = 2 - c$, where c is the number of collinear adjacent edges. The value for c is taken from a matrix, which contains the relative angles between the bitangent edges and previous/next edges on the obstacles. If, as depicted in the upper part of Figure 4.7, no adjacent edge of e_5 is collinear, then the condition to determine the visibility is $k_{bt} = 2 - 0 = 2$ (depicted by two purple dots). Otherwise, as shown in the middle, one edge (e_2) is collinear, the condition for visibility is $k_{bt} = 2-1 = 1$ (depicted by one purple dot). Finally, if as shown in the lower part of Figure 4.7, both adjacent edges of e_2 are collinear (e_1, e_3), visibility is provided if $k_{bt} = 2-2 = 0$ (no purple dot).

Also for bitangent edges, whose two vertices do not belong to the same obstacle, three cases have to be distinguished, in order to determine their visibility. As both vertices are on different obstacles, the maximum number of intersecting edges is four.

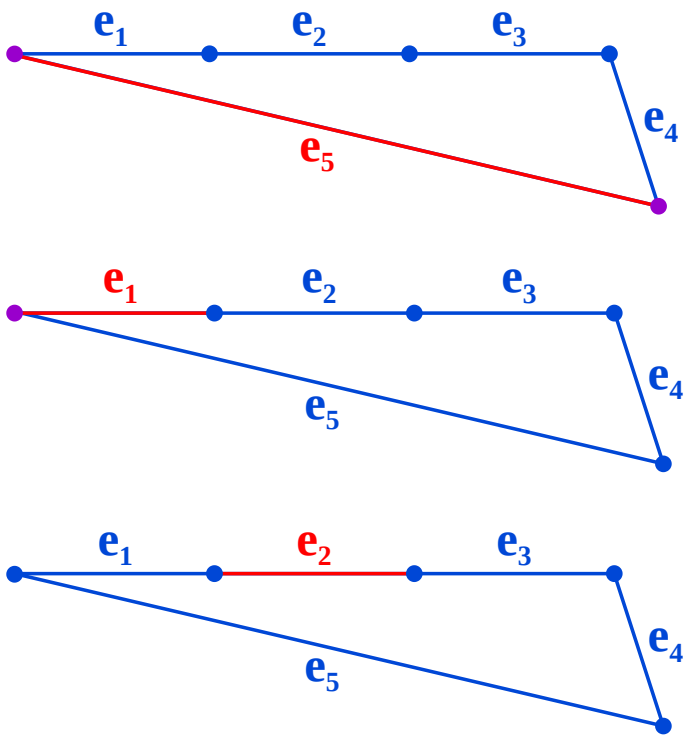


Figure 4.7: *For bitangent body edges three cases have to be distinguished. Either none, one or both adjacent edges can be collinear. This is decisive for the number of required intersections, which are depicted by purple dots.*

However, if a bitangent edge is collinear with an edge of one or both obstacles, the number of required intersections for visibility has to be adapted. The rule to overcome degeneracies is $k_{bt} = 4 - c$, where c is the number of collinear adjacent edges. In the upper part of Figure 4.8, no edge is collinear to the red bitangent edge and therefore $k_{bt} = 4 - 0 = 4$. In the middle, the edge e_6 is collinear with the red bitangent edge and therefore $k_{bt} = 4 - 1 = 3$. In the lower part, two collinear edges (e_3, e_6) result in $k_{bt} = 4 - 2 = 2$. All three bitangent edges in the example are visible.

In the example in Figure 4.9, a special case with degeneracies is presented, in which the initial state x_{in} and four vertices v_2, v_5, v_6 and v_8 of the polygonal obstacles are collinear. It is tested, whether a visible connection between x_{in} and v_8 can be established, using the aforementioned conditions for the number of intersections. The red edges are tangent and bitangent edges $\in E_R$, while the obstacle edges $\in E_R$ are depicted in blue. Purple dots mark points of intersection between the red and the blue edges. First, in upper part of Figure 4.9, the direct tangent edge from the initial state x_{in} to vertex v_8 is checked for visibility. The number of admissible intersections is $k_t \leq 2$. It is intersected by the edges e_1, e_2, e_4, e_6, e_7 and e_8. The number of intersections is $k_{bt} = 6$ and thus the edge is invisible. In the middle, the tangent edge from x_{in} to v_2 is visible, as it is only

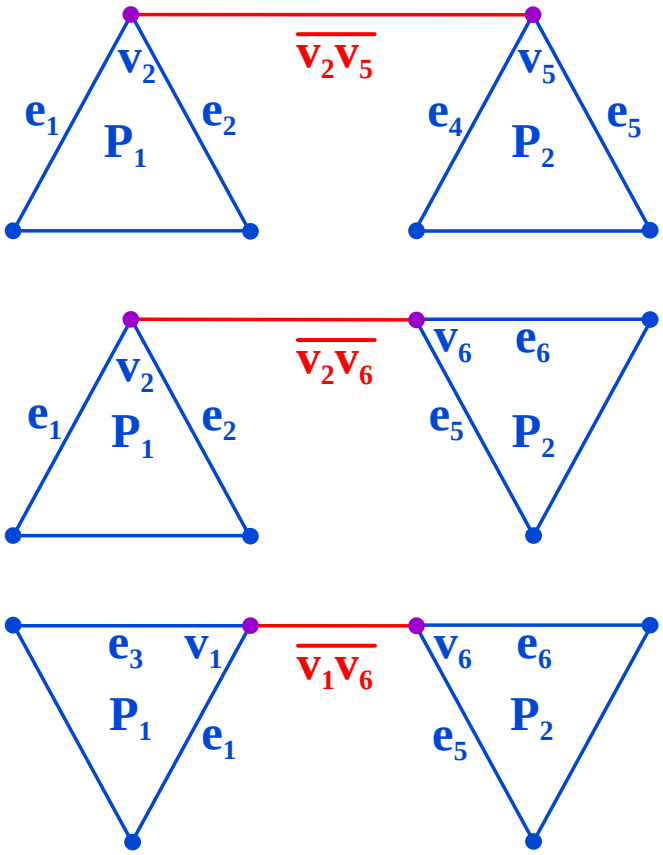


Figure 4.8: *Three different cases for bitangent edges, connecting two obstacles. Either none, one or two, out of four adjacent edges, can be collinear. For the determination of visibility, the number of collinear edges is retrieved.*

intersected by e_1 and e_2 and the number of intersections is $k_t = 2$. It is also attempted to establish the connection to v_8 with the direct bitangent edge from v_2, which is intersected by the edges e_1, e_2, e_4, e_6, e_7 and e_8. Therefore, the number of intersections is $k_{bt} = 6$ and the edge is invisible. In the lower part of Figure 4.9, the visible connection between x_{in} and v_8 is established, using intermediate connections. The tangent edge from x_{in} to v_2 ($k_t = 2$) is followed by the bitangent edge from v_2 to v_5. As the upper edge of P_2 is collinear to it, $c = 1$ and therefore $k_{bt} = 4 - c = 3$. As v_2 to v_5 is intersected by the edges e_1, e_2 and e_4, the edge is visible. The edge from v_5 to v_6 is a bitangent body edge, which is why $k_{bt} = 2 - c$ applies. As the adjacent edges e_4 and e_6 are not collinear $k_{bt} = 2 - 0 = 2$, which is why it is visible. Finally, the bitangent edge from v_6 to v_8 is again collinear with the upper edge of P_2 and therefore $k_{bt} = 4 - 1 = 3$. It is intersected by the edges e_6, e_7, e_8 and is therefore visible. This proves that the conditions for the number of intersections are sufficient to compute a reduced visibility graph VG_r.

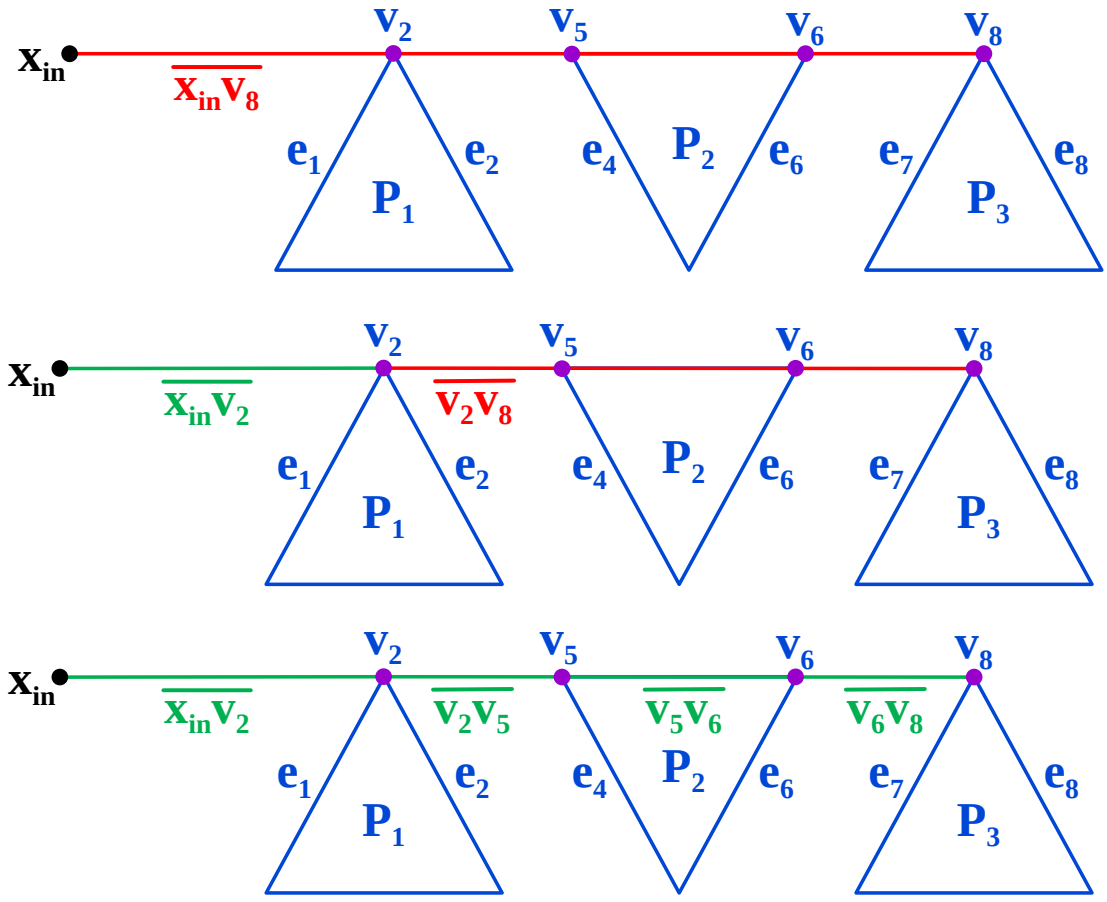


Figure 4.9: *This example shows, how the connectivity can be established, even in the special case of several collinear vertices/edges.*

Partial Shortest Path Map

In order to find the shortest path from start to goal, it is not necessary to construct a complete visibility graph. Instead [58] presented the continuous Dijkstra paradigm, which effectively computes a shortest path map (SPM), running in $O(|V| \log |V|)$ [58, 59, 93, 9]. To further speed up the search, a new approach is presented. The idea is to avoid the construction of a complete SPM and instead build a partial shortest path map (PSPM). For the presented approach, the A*-search algorithm (Chapter 4.2) is applied, which is optimal efficient. This means that no other known algorithm expands its search less, using the same heuristic function [102]. As estimator for the cost-to-go the Euclidean distance is applied, which is a consistent heuristic, as it obeys the triangle inequality [103]. As A* is an informed search algorithm, the number of examined edges is generally significantly lower than with Dijkstra, which expands like a wavefront. By searching for the optimal solution to the goal, a PSPM is iteratively assembled from individual partial visibility graphs VG_p, until the goal is connected to the start via visible edges.

A VG_p exclusively contains tangent edges, between an initial state and its adjacent states. The selection of tangent edges takes linear time $O(|V_{EX}|)$. As explained in Section 4.1.2, the visibility of edges can be determined solely by the number of intersections with the obstacle edges. This can be done with the algorithm given in [100] (see Section 4.1.2) in $O(|E_t + E_{EX}|^{4/3} \log |E_t + E_{EX}|)$. Alternatively, a rotational sweep line segment algorithm [95] can be applied to determine the visibility, which runs in $O(|V_{EX}| \log |V_{EX}|)$. Checking if an edge is tangent takes constant time $O(|V_{EX}|)$. Therefore, the time complexity of visible tangent edges, in each iteration of the A*-search, takes $O(|V_{EX}| \log |V_{EX}|)$, which is also the overall time complexity. The only nontangent edge, which is always checked first for visibility, is the direct edge between the initial state and the goal state. The sweep starts from the angle of this edge and performs a 2π rotation, if the goal is visible the algorithm breaks and the PSPM is constructed.

The worst time complexity of A*-search is that of Dijkstra's algorithm, which is the case when no heuristic is applied. If A*-search is applied with a consistent heuristic (see Section 4.3 and 4.3.2), a PSPM from start to goal is constructed (see Figure 4.10(d)) and searched at the same time. For this example, as few as $|E_t| = 124$ tangent edges were examined (in two iterations), of which 71 are visible. Applying this procedure results in significant computational savings, while both completeness and optimality of the search are guaranteed. This method works for static as well as dynamic environments and is applied in Sections 4.3.1 and 4.3.2 to find time-minimum trajectories amidst moving obstacles. The shortest map from start to goal amidst time-varying obstacles is called the partial shortest trajectory map (PSTM), which is applied for trajectory planning in Section 4.2 and Chapter 6.

Visibility Graph Examples

In general, a visibility graph is computed for an estimated state space EX, which according to Equation (3.7), is the difference between the workspace and the estimated obstacle region (Section 3.3)

$$W \setminus EX_{obs}.$$

For better visualization and reproducibility, in the following example, the obstacle region EX_{obs} is given by a static set of polygonal obstacles O from the Rad-TRAM nowcast, issued on 2015/07/07 at 14 : 38 h. The coordinates of the mission area W are 45.00° to 55.00°, in latitude, and 3.00° to 16.00°, in longitude. In the presented example, the polygons of the state space have $|V_{EX}| = 513$ unique vertices and $|E_{EX}| = 541$ unique

edges. The state space corresponds to the one shown in Figure 4.10(a), where in this case, an unreduced VG is presented. In (b), a VG_r is calculated for the line-smoothed state space. With the same settings a VG_r in a limited corridor is shown in (c). Finally in (d), a partial shortest path map is depicted. According to Equation (4.2) the number of tangent edge candidates, in this example, is

$$|E_{tc}| = 2 \cdot 513 = 1026 \tag{4.7}$$

and according to Equation (4.3) the number of unique bitangent edge candidates E_{btc} is

$$|E_{btc}| = \frac{|V_{EX}|^2 - |V_{EX}|}{2} = \frac{|513|^2 - |513|}{2} = 131328. \tag{4.8}$$

The total runtime for the selection of tangent and bitangent edges, as a function of the number of vertices (including start and goal), is plotted in Figure 4.11. The vectorized Matlab R2015b code runs on a computer with Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz, 32GB RAM and 64-Bit-Windows 7.

The resulting number of tangent edges in the example is $E_t = 154$. According to Equation (4.6), the sparsity of the tangent edge matrix M_t is

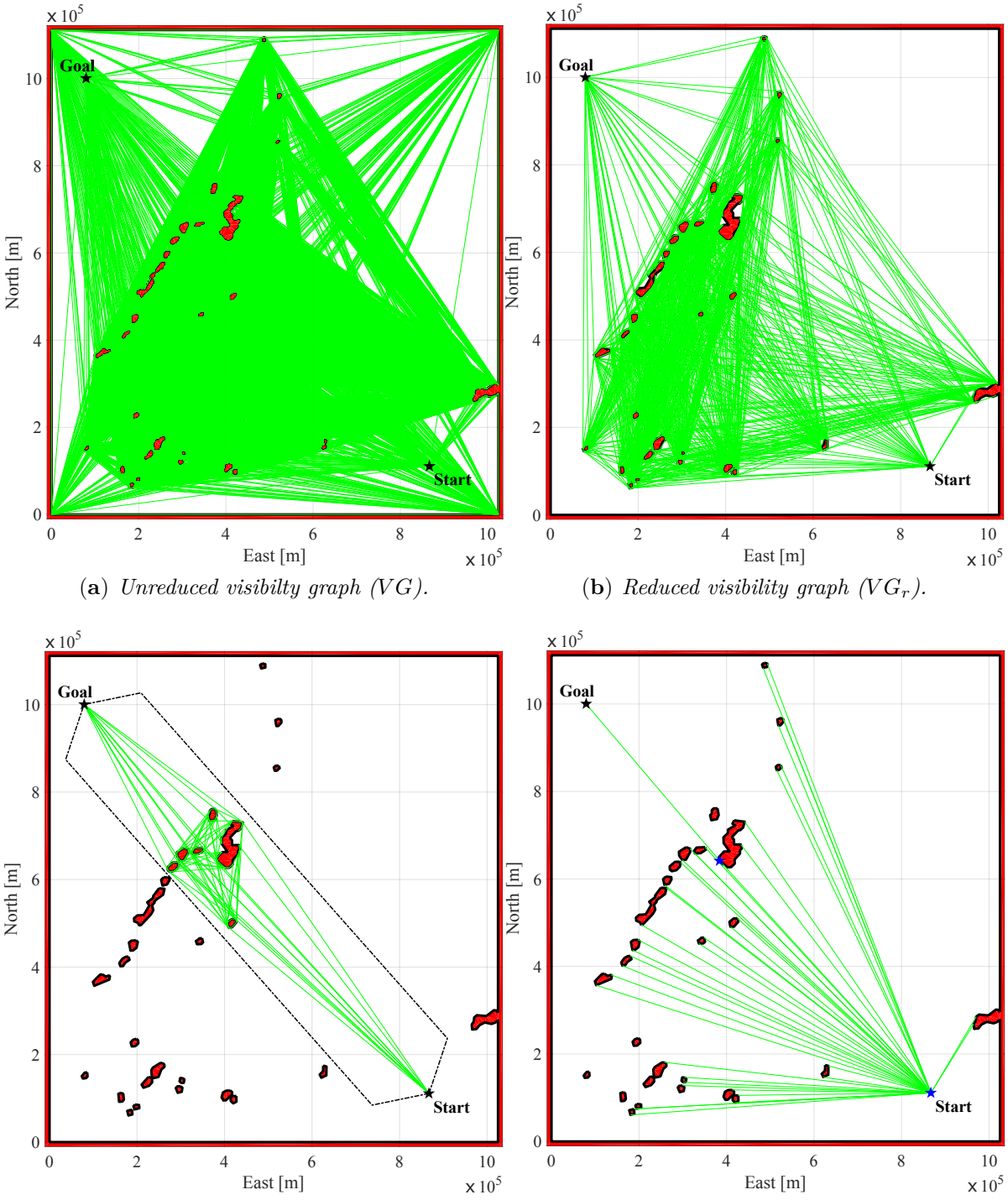
$$S(M_t) = \frac{E(M_t) - Z(M_t)}{E(M_t)} 100 = \frac{1026 - 154}{1026} 100 = 84.96\ \%. \tag{4.9}$$

The maximum number of unique edges, between the obstacles, using Equation 4.3, is $|E_{btc}| = 131328$. The resulting number of bitangent edges in the example is $|E_{bt}| = 5832$. According to Equation (4.6), the sparsity of the bitangent matrix M_{bt} is

$$S(M_{bt}) = \frac{E(M_{bt}) - NZ(M_{bt})}{E(M_{bt})} 100 = \frac{131328 - 5832}{131328} 100 = 95.56\ \%. \tag{4.10}$$

By definition, M_t and M_{bt}, are both sparse. Figure 4.12 shows the sparsity of M_{bt}, which contains $E_{bt} = 5832$ unique bitangent edges, as a combination of the vertices in the estimated state space V_{EX}.

If nontangent edges are not filtered, the number of visible edges from start and goal to the obstacles is 363. The number of visible tangent edges is only $|E_{vt}| = 85$, which corresponds to a reduction of 76.85 %. If nonbitangent edges are not filtered, the number of visible edges, between the obstacle polygons of EX, is 17965, in Figure 4.10(a). The number of visible bitangent edges is only $E_{vbt} = 1668$, which corresponds to a 90.72 % reduction.



(**a**) *Unreduced visibilty graph (VG).*

(**b**) *Reduced visibility graph (VG_r).*

(**c**) *Reduced visibility graph (VG_r), in a corridor.*

(**d**) *Partial shortest path map with A*-search.*

Figure 4.10: *The unreduced VG, in (a), contains* 18842 *visible edges. The reduced complete VG_r with line-smoothing ($\epsilon = 0.022°$), in (b), contains* 1273 *visible edges, which is a* 93.24 *% reduction compared to (a). In (c) the mission area is limited to a corridor with* 230 *km of width and an opening angle at start and goal of* 120°*. This reduces the total number of visible edges to* 131*, while the optimal trajectory is included (which is not guaranteed, if the corridor is too narrow). The partial shortest path map (PSPM), in (d), is assembled from two initial states (blue stars), performing an A*-search with Euclidean distance heuristic. The number of visible edges is reduced to* 41*, while the optimal solution is guaranteed to be included, if the applied heuristic is monotone.*

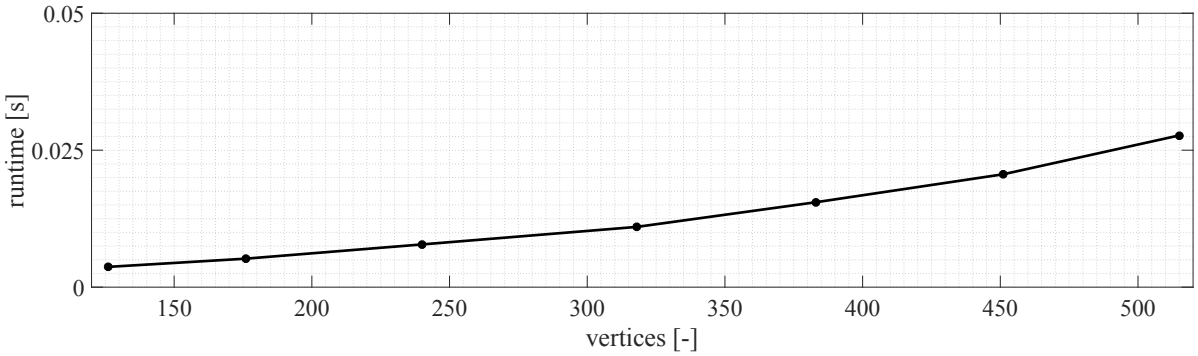


Figure 4.11: *Seven degrees of line-smoothing are applied to the scenario in Figure 4.10(a). Runtimes for tangent and bitangent edge selection, as a function of varying number of vertices $|V_{EX}|$ (due to different degrees of line-smoothing), are depicted with black dots.*

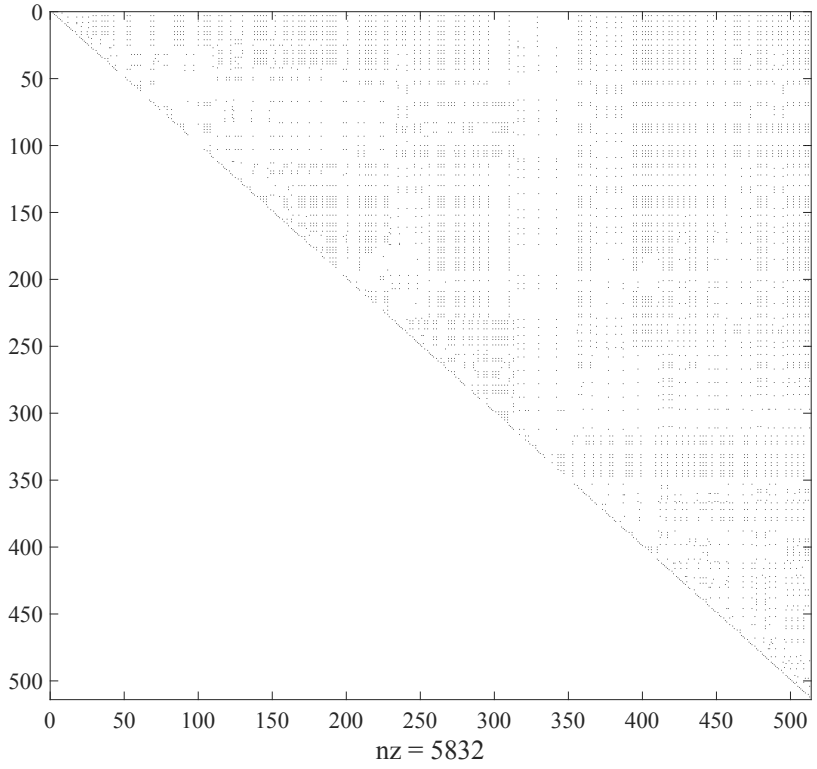


Figure 4.12: *Sparse adjacency matrix of unique bitangent edge candidates in EX. The number of nonzero elements is 5832. The sparsity of the upper triangular matrix is* 95.56 *%.*

Tables 4.2 to 4.9 compare the four methods shown in Figure 4.10. These are the complete unreduced (a), complete reduced (b) and incomplete reduced visibility graph (c), as well as the partial shortest path map (d). For the following considerations, the cost for the computation of tangent and bitangent edges is not accounted. Several parameters are computed, among others the worst case search cost of an ensuing graph-search with Dijkstra's algorithm. The time-complexity of Dijkstra's algorithm is $O((|E| + |V|) \log |V|)$,

if a binary min-heap is applied for the priority queue. According to [104] the worst-case bound of Dijkstra, when using a Fibonacci heap, is $O(|E|+|V|\log|V|)$. Filtering nontangent and nonbitangent edges from the visibility graph, implicitly reduces the number of vertices in the visibility graph. In the presented example, the number of unique vertices is reduced from initially $|V_{EX}| = 513$, by 28.46 %, to $|V_{VG_r}| = 367$. The sum of all edges in the VG_r is

$$E_{VG_r} = E_{vt} + E_{vbt} = 85 + 1668 = 1753. \tag{4.11}$$

The number of operations during a Dijkstra search on the reduced graph VG_r is therefore

$$OP_{Dr} = |E_{VG_r}| + |V_{VG_r}| \log |V_{VG_r}| = 1753 + 367 \log 367 = 1753 + 3127 = 4880. \tag{4.12}$$

Whereas, the unreduced VG contains 363 edges, from start and goal states to the obstacles, 17965 edges, between the obstacles, and $E_{EX} = 541$ obstacle edges, which adds up to a total of $E_{VG} = 363 + 17965 + 514 = 18869$ visible edges. The number of unique vertices in the VG is equal to $|V_{EX}| = 513$. The number of operations during a Dijkstra search on the unreduced graph VG is

$$OP_D = |E_{VG}| + |V_{VG}| \log |V_{VG}| = 18842 + 513 \log 513 = 18869 + 4618 = 23487. \tag{4.13}$$

The ratio of the number of operations from Equations (4.12) and (4.13) allows to estimate the reduction w.r.t. to the Dijkstra search. SC is defined as the worst case search cost to find the shortest path in a visibility graph with Dijkstra's algorithm, relative to the unreduced VG (without line-smoothing). In this case, the search of the reduced visibility graph VG_r reduces the maximum number of operations to

$$SC = \frac{OP_{Dr} \cdot 100}{OP_D} = \frac{4880 \cdot 100}{23487} = 20.78\ \% \tag{4.14}$$

compared to a search of the unreduced visibility graph VG.

Applying line-smoothing on the estimated state space EX, before a visibility graph is computed, is beneficial for unreduced and reduced variants, as the number of vertices $|V_{EX}|$ is reduced. Furthermore, a variant with a limitation of the workspace/mission area is compared. Four different visibility graph variants are compared in the following, using ten different estimated state spaces for each. The results are for setups, without (Tables 4.2, 4.3 and 4.4) and with line-smoothing (Tables 4.6, 4.7 and 4.8), including

variants with restricted workspace W (Tables 4.4 and 4.8). In the following tables, $|V_{EX}|$ is the number of unique vertices in the estimated state space EX. If line-smoothing is applied to EX, $|V_{LS}|$ is the resulting number of unique vertices, in the smoothed EX. The final number of vertices, in the resulting visibility graph, is listed as $|V_{VG}|$. $|E_{VG}|$ is the resulting number of visible edges, in a visibility graph. In Chapter 4.2, the visibility graph is searched by an A*-search algorithm, whose worst time complexity is $O(b^d)$. The base is the so called branching factor b and the exponent d is the depth of the final solution. Hence, the mean branching factor $\bar{b}$ has an important influence on the convergence of the search, which is why it is additionally listed. It is calculated by

$$\bar{b} = \frac{2|E_{VG}|}{|V_{VG}|}. \tag{4.15}$$

As $\bar{b}$ represents the mean number of connected edges/states, the results are rounded to the next integer for better intelligibility. Using a VG_r, which exclusively contains tangent and bitangent edges, significantly reduces the mean branching factor. On average, in Table 4.2, the mean branching factor is $\bar{b} = 68$, for the unreduced visibility graph, while it is $\bar{b} = 8$ in Table 4.3, for the reduced one.

Table 4.2: *Unreduced visibility graph, on data from the 2015/07/07, without line-smoothing.*

Time	$\|V_{EX}\|$	$\|V_{LS}\|$	$\|V_{VG}\|$	$\|E_{VG}\|$	$\bar{b}$	**SC** [%]
12:00	241	-	-	6867	57	100
13:00	461	-	-	13742	60	100
14:00	341	-	-	11397	67	100
15:00	669	-	-	24930	75	100
16:00	672	-	-	25478	76	100
17:00	808	-	-	28918	72	100
18:00	912	-	-	31565	69	100
19:00	1196	-	-	42996	72	100
20:00	996	-	-	34945	70	100
21:00	1061	-	-	34868	66	100
22:00	994	-	-	32734	66	100

Without line-smoothing, the differences between the results in Table 4.2, for the unreduced, and Table 4.3, for the reduced visibility graph, regarding the ensuing search with Dijkstra's algorithm, are considerable. By applying line-smoothing in the following examples, the difference becomes less pronounced, but is still considerable.

While the branching factor can be influenced significantly by line-smoothing in the unreduced VG (compare $\bar{b}$, in Tables 4.2 and 4.6), for the reduced versions mainly the

Table 4.3: *Reduced visibility graph, on data from the 2015/07/07, without line-smoothing. The ratio between the initial number of vertices $|V_{EX}|$ in the estimated state space is reduced to $|V_{VG}|$. Implicitly the obstacle edges are likewise reduced by* 29.61 *%.*

Time	$\boldsymbol{\lvert V_{EX}\rvert}$	$\boldsymbol{\lvert V_{LS}\rvert}$	$\boldsymbol{\lvert V_{VG}\rvert}$	$\boldsymbol{\lvert E_{VG}\rvert}$	$\bar{\boldsymbol{b}}$	***SC*** [%]
12:00	241	-	204	802	8	26.98
13:00	461	-	341	1441	8	24.18
14:00	341	-	269	1234	9	23.87
15:00	669	-	478	2216	9	20.73
16:00	672	-	484	1996	8	19.86
17:00	808	-	573	2367	8	20.74
18:00	912	-	614	2383	8	19.91
19:00	1196	-	833	3302	8	20.61
20:00	996	-	702	3073	9	21.64
21:00	1061	-	704	2750	8	20.66
22:00	994	-	676	2709	8	21.26

Table 4.4: *Reduced visibility graph, on data from the 2015/07/07, without line-smoothing, in a limited corridor.*

Time	$\boldsymbol{\lvert V_{EX}\rvert}$	$\boldsymbol{\lvert V_{LS}\rvert}$	$\boldsymbol{\lvert V_{VG}\rvert}$	$\boldsymbol{\lvert E_{VG}\rvert}$	$\bar{\boldsymbol{b}}$	***SC*** [%]
12:00	46	-	40	55	3	3.05
13:00	27	-	19	16	2	0.54
14:00	80	-	56	69	2	2.76
15:00	230	-	143	203	1	3.93
16:00	161	-	108	143	2	2.75
17:00	175	-	115	147	2	2.54
18:00	174	-	114	115	2	2.21
19:00	245	-	168	283	3	2.76
20:00	387	-	255	410	3	5.46
21:00	331	-	204	232	2	3.95
22:00	299	-	191	284	3	4.06

number of obstacles in the workspace is decisive. This is the reason why the branching factor in the example with the restricted workspace (Tables 4.4 and 4.8) is generally lower, than in the unrestricted case (Tables 4.3 and 4.7). Therefore, it can be summarized, that the smoothing of obstacles, a reduced visibility graph as well as a reduction in the number of obstacles (resulting from a restriction of the search space) have a significant influence on the runtime of the trajectory planning.

The combination of line-smoothing and reduced visibility graph (see Table 4.7) usually results in considerable reductions of computational effort. Table 4.8 shows, how an additional restriction of the workspace W offers further potential for savings. However, the probability of finding a trajectory is reduced and optimality can no longer be ensured.

Table 4.5: *Partial shortest path map, on data from the 2015/07/07, without line-smoothing. The number of computed visible edges, which are necessary to find the shortest path to the goal, is relatively small. Generally, a partial shortest map is the most effective and fastest of all presented visibility methods.*

Time	$\|\boldsymbol{V_{EX}}\|$	$\|\boldsymbol{V_{LS}}\|$	$\|\boldsymbol{E_{VG}}\|$	**A* Iter.**
12:00	241	-	37	2
13:00	461	-	1	1
14:00	341	-	1	1
15:00	669	-	16	1
16:00	672	-	97	5
17:00	808	-	122	4
18:00	912	-	42	2
19:00	1196	-	104	4
20:00	996	-	180	11
21:00	1061	-	160	22
22:00	994	-	73	9

Table 4.6: *Unreduced visibility graph, on data from the 2015/07/07, with line-smoothing. A value of $\epsilon = 0.022°$, more than halves the original number of vertices ($|V_{EX}|$ vs. $|V_{LS}|$).*

Time	$\|\boldsymbol{V_{EX}}\|$	$\|\boldsymbol{V_{LS}}\|$	$\|\boldsymbol{V_{VG}}\|$	$\|\boldsymbol{E_{VG}}\|$	$\bar{\boldsymbol{b}}$	***SC*** [%]
12:00	241	135	-	2967	44	44.70
13:00	461	236	-	5603	47	41.88
14:00	341	189	-	4846	51	43.99
15:00	669	310	-	7815	50	33.26
16:00	672	322	-	7769	48	32.88
17:00	808	380	-	9147	48	33.78
18:00	912	417	-	10297	49	34.36
19:00	1196	564	-	14721	52	35.99
20:00	996	483	-	12469	52	37.39
21:00	1061	448	-	9933	44	30.48
22:00	994	453	-	10894	48	34.93

The restriction of the workspace can also be on the basis of a distinction between broad and near phase. The workspace is then reduced to the area of the reachable set of future aircraft states A_{fut} (see Section 3.3) in the prediction horizon T_N of the nowcast. This restriction of the workspace is applied in all examples in Chapter 6.

Table 4.7: *Reduced visibility graph, on data from the 2015/07/07, with line-smoothing.*

Time	$\|V_{EX}\|$	$\|V_{LS}\|$	$\|V_{VG}\|$	$\|E_{VG}\|$	$\bar{b}$	SC [%]
12:00	241	135	127	690	11	17.98
13:00	461	236	198	1100	11	14.65
14:00	341	189	168	1003	12	15.74
15:00	669	310	263	1468	11	11.48
16:00	672	322	278	1351	10	11.35
17:00	808	380	321	1553	10	11.51
18:00	912	417	335	1537	9	10.81
19:00	1196	564	475	2305	10	11.82
20:00	996	483	397	2182	11	12.50
21:00	1061	448	363	1623	9	10.34
22:00	994	453	364	1781	10	11.44

Table 4.8: *Reduced visibility graph, on data from the 2015/07/07, with line-smoothing, in a limited corridor.*

Time	$\|V_{EX}\|$	$\|V_{LS}\|$	$\|V_{VG}\|$	$\|E_{VG}\|$	$\bar{b}$	SC [%]
12:00	46	35	29	55	4	2.23
13:00	27	20	12	16	3	0.33
14:00	80	47	35	49	3	1.61
15:00	230	117	78	148	4	2.04
16:00	161	80	63	115	4	1.55
17:00	175	82	60	84	3	1.19
18:00	174	83	62	96	3	1.15
19:00	245	129	103	209	4	1.63
20:00	387	172	134	286	4	2.75
21:00	331	138	99	153	3	1.78
22:00	299	137	97	186	4	1.94

Table 4.9: *Partial shortest path map, on data from the 2015/07/07, with line-smoothing. The number of computed visible edges, which are necessary to find the shortest path to the goal, is relatively small.*

Time	$\|V_{EX}\|$	$\|V_{LS}\|$	$\|E_{VG}\|$	**A* Iter.**
12:00	241	135	36	2
13:00	461	236	1	1
14:00	341	189	1	1
15:00	669	310	1	1
16:00	672	322	80	4
17:00	808	380	82	3
18:00	912	417	39	2
19:00	1196	564	77	3
20:00	996	483	153	16
21:00	1061	448	70	7
22:00	994	453	37	3

4.2 Informed Search in Dynamic Environments

In order to find anticipatory near-time-optimal trajectories, A*-search is applied [53]. It is suitable for single-source-single-target problems and has many advantageous properties, namely being a complete, optimal, optimal efficient and easy to implement algorithm. The contributions in this chapter are the concept of partial shortest trajectory maps, changes to the A*-search algorithm when applied to exclusively moving obstacles (Section 4.2.2) and an effective heuristic to speed up the search (Section 4.3.3).

4.2.1 A*-Search for Static Environments

The A* is a straightforward informed search algorithm, which means that it uses domain knowledge to reduce the amount of search in order to find an optimal solution. If the applied heuristic is at least admissible (see Section 4.3), it is guaranteed to find the optimal solution, e.g. the shortest path. Furthermore, if a consistent heuristic is used (see Section 4.3), A*-search is provably optimal efficient, which means that no other known algorithm, using the same heuristic, expands fewer nodes in order to find a solution [102]. It is also a complete algorithm, which means that if a solution exists it is found.

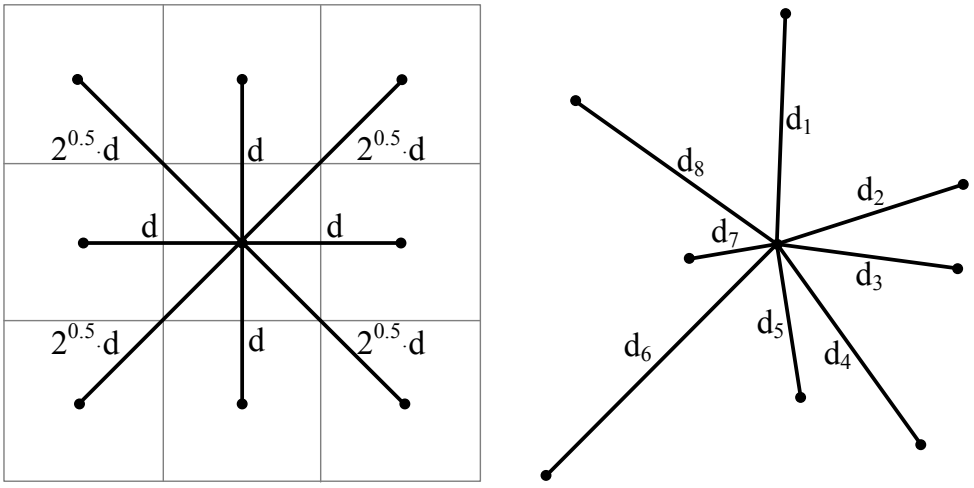


Figure 4.13: *Analogous to the equidistant grid on the left and a nonhomogeneous graph, which represents a visibility graph (see Section 4.1), is depicted on the right side. In contrast to an indifferent raster discretization it exclusively contains the actually achievable shortest paths (straight lines). In conjunction with the methods presented in Section 4.1 this leads in total to a reduction of the search, which represents a significant contribution of this work.*

The algorithm is often applied to a grid representation of the search space. Typical grids are triangular, square or hexagonal. Left in Figure 4.13, a square grid is depicted, which allows movement in eight directions, where the cost of a straight movement is the

distance d and the diagonal cost is $\sqrt{2}d$. Depending on the resolution of a grid, the search can be rather slow, as a great deal of intermediate nodes have to be expanded. To speed up the search on uniform-cost grid maps, jump point search can be applied [105]. For the present application a visibility graph with heterogeneous, nonnegative edge weights is used, as depicted on the right in Figure 4.13. A reduced visibility graph contains no unnecessary vertices and edges (see Section 4.1.2) and innately allows to cover greater distances without intermediate nodes.

In Algorithm 1 the basic A*-search for a static environment is described. The start configuration is q_s and the goal configuration is q_g. Initial or intermediate configurations are q_{in}, adjacent configurations to q_{in} are q_a. The bookkeeping is done in the so called open list (OL). Configurations, which are stored in OL are called candidates q_c. The total estimated F-cost is the distance it takes, to get from the start configuration q_s to goal configuration q_g, via an adjacent configuration q_a, and is calculated by

$$F(q_a) = G(q_a) + H(q_a), \tag{4.16}$$

where G is the actual cost to get from the start to an adjacent configuration and H is the estimated cost to get from an adjacent to the goal configuration. If an initial configuration q_{in} is the goal configuration q_g (see line 18, in Algorithm 1) and the applied heuristic is at least admissible, the shortest path is computed.

For static environments the roadmap and even the heuristic costs can be preprocessed. Alternatively, in Section 4.1.2, the A*-search is applied to compute the roadmap of the environment, while searching the shortest path to a goal. The resulting graph is termed partially shortest path map (PSPM). In this case, in Algorithm 1, a function is called between line three and four, which computes the visible tangent edges from the actual initial configuration q_{in}, to determine the q_a. Each time the visibility of the nontangent edge between q_{in} and q_g is additionally determined.

4.2.2 A*-Search for Dynamic Environments

This section features Contribution 4: *A*-Search in Dynamic Environments - Partial Shortest Trajectory Map.*

A free estimated state space EX_{free} is computed by the prediction unit (Section 3), whereof different kinds of search-graphs can be created, using the methods from Section 4.1. In this section the associated informed graph search is presented. The A*-search

Algorithm 1: Basic A*-search

Input: start, goal, roadmap, heuristic values
Output: shortest path

initalize open list $OL = \{q_s\}$ with the start as initial configuration
initalize empty closed list $CL = \emptyset$
while $OL \neq \emptyset$ **do**
  foreach q_a *from the actual* q_{in} **do**
    if $q_a \notin CL$ **then**
      calculate total cost from q_s to q_g via q_a with $F(q_a) = G(q_a) + H(q_a)$
      if $q_a \notin OL$ **then**
        insert q_a as q_c in OL
      else
        if *new* $G(q_c) <$ *than prior* $G(q_c)$ **then**
          replace existing G-cost and set q_{in} as parent configuration q_p
        end
      end
    end
  end
  pick $q_c \in OL$ with the minimum F-cost as new q_{in}
  remove q_{in} from the OL and insert it into CL
  if q_{in} *is the goal state* q_g **then**
    break
  end
end
if $OL = \emptyset$ **then**
  return no path found
else
  construct path by backtracking of the parent nodes
end

algorithm from Section 4.2.1 is adapted, in order to compute near-optimal trajectories, regarding the prediction from Chapter 3. In the upcoming Section 4.3 a new heuristic function is introduced to speed up the search.

Prior research, regarding the computation of optimal trajectories in the presence of time-varying obstacles, was done by [39], using visibility graph representation of the search space. Time-minimal motion to a moving destination point can be computed, for the case, in which the agent moves faster than any of the obstacles. The exclusively convex polygonal obstacles move in a fixed direction at constant bounded velocity.

A heuristic search algorithm for motion planning amidst moving obstacles is presented in [40]. Safe time-minimal trajectories are computed for the special case of growing discs with bounded velocity. The agent is required to move faster than any of the obstacles. Cyclic replanning of trajectories and online application, including a compensation of the planning time, are discussed.

In both approaches, the obstacles are exclusively convex and nonholonomic constraints are not considered, although [40] computes smooth paths by using round obstacles. In the presented case, obstacles are represented by arbitrary polygons, for whose future state a prediction exists. Aircraft and obstacles move at bounded velocity, without a relative velocity limitation, i.e. an obstacle is allowed to move faster than the aircraft. In order to be able to consider dynamic environments, the configurations q are parameterized by time t. Thus, the A*-search operates with states $\mathrm{x} = (q, t)$ instead of configurations q.

At the beginning of the algorithm, the only state in the OL is x_s, which automatically has the lowest estimated F-cost. The $EX_{free}(\mathrm{x}_{in})$ is computed from $\mathrm{x}_\mathrm{s} = \mathrm{x}_{in}$. Depending on the applied heuristic a partial or complete visibility representation of $EX_{free}(\mathrm{x}_{in})$ is generated, whose nodes are treated as states. In addition to the determination of the adjacent states to the initial state, either a complete reduced visibility graph or partial shortest path map are used to estimated the heuristic costs (see Section 4.3.3). An adjacent state x_a is directly connected to its parent state $\mathrm{x}_p = \mathrm{x}_{in}$, via a visible edge. For each initial state, at least one so called auxiliary states x_{aux} is added, as adjacent state. This is necessary, to simulate the nonholonomic turning-flight constraints of a fixed-wing aircraft, which is explained in detail in Chapter 5.

In the presented approach, a complete roadmap to the goal in X-space does not exist a priori. Instead, it is incrementally generated by exploring initial states x_{in_i}, with $i \in \{1, \ldots, k\}$. This results in a partial shortest trajectory map (PSTM) to the goal state, which is analogous to the partial shortest path map (SPM) from the previous section. For every x_{in}, an estimated state space $EX_{free}(\mathrm{x}_{in})$ is computed and the adjacent states

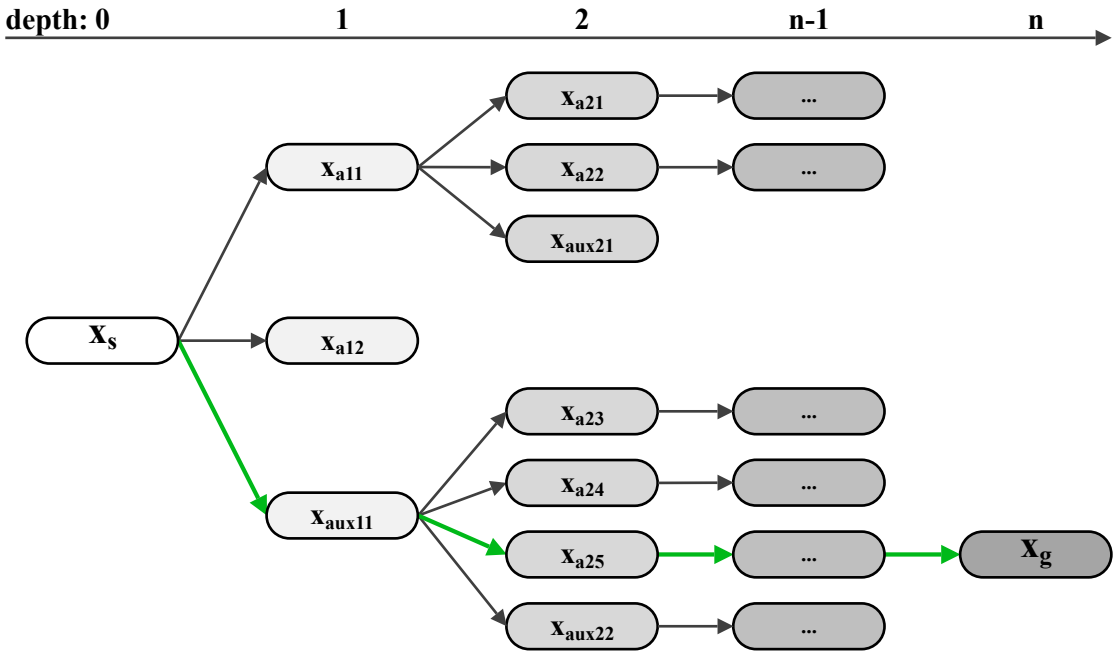


Figure 4.14: *Schematic of a partial shortest trajectory graph. Adjacent states (e.g. depth 1 states) are visible in the free estimated state space EX_{free}, from an initial state (e.g. start state, depth 0). The A*-search always expands the most promising states, from which a new EX_{free} is computed, in order to determine its adjacent states. Additionally at least one auxiliary state is added. Incrementally, a partial shortest trajectory map is built, until the goal state is reached (green arrows).*

are determined. By doing so, A*-search generates a tree of visible connections, which is rooted in the start state x_s (see Figure 4.14). This process is repeated, until the goal state x_g is the initial state x_{in} or no solution exists.

In contrast to the Dijkstra's algorithm, A*-search does not just compute the G-cost, which in this case is the time to get from the start state x_s to an adjacent state x_a. Instead, it uses an additional heuristic, to estimate the H-cost, which is the time to get from x_a to x_g. As in the basic A*-search algorithm, the total estimated F-cost is the time it takes, to get from x_s to x_g, via x_a, and is calculated by

$$F(\mathrm{x}_a) = G(\mathrm{x}_a) + H(\mathrm{x}_a). \tag{4.17}$$

In every while loop of the A*-search, the F-cost for each x_a is computed. If the condition in Algorithm 2, line 20 is met, an x_a is inserted into OL. States in OL, are called candidate states x_c. The OL is a priority queue, which is implemented as binary heap. It contains the candidate states, sorted in ascending F-cost order, as well as their respective parent states. In order to compute time-monotone and feasible trajectories, for each x_c additional state information is stored in OL (see Table 4.10).

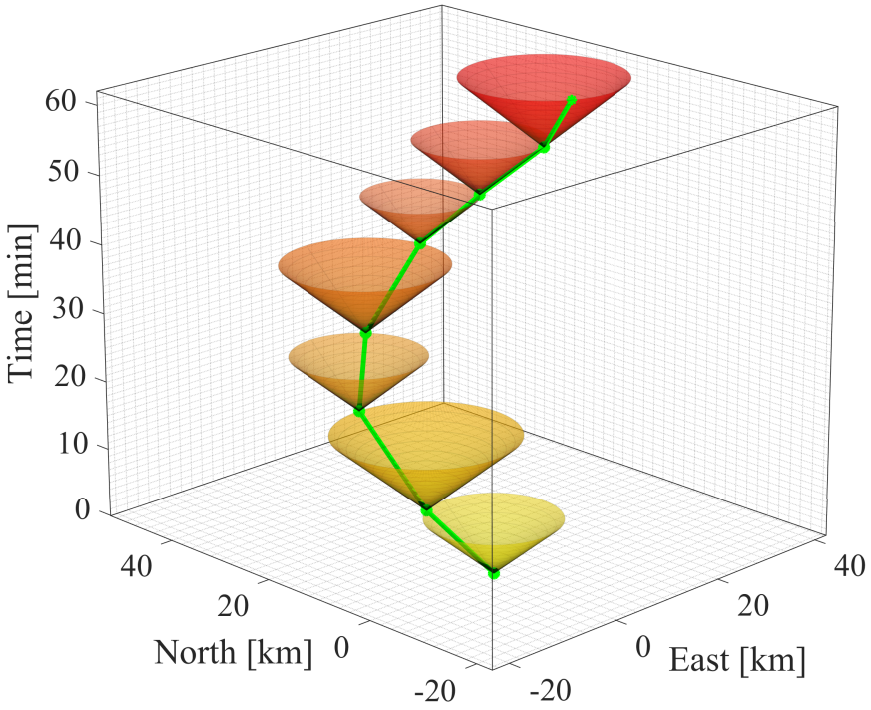
Table 4.10: *The open list stores various information about its candidate states. If a candidate state is set as new initial state, the information is retrieved, to compute a feasible trajectory. The time-at-arrival* $t(\mathrm{x}_c)$ *ensures, that a trajectory is time-monotone, while course-at-arrival* $\chi(\mathrm{x}_c)$ *and bank-at-arrival* $\mu(\mathrm{x}_c)$ *are necessary, to constrain the curvature of the trajectory (see Chapter 5).*

Open List Information of Candidate States		
1	$\boldsymbol{x}(\mathbf{x}_c)$	x-coordinate
2	$\boldsymbol{y}(\mathbf{x}_c)$	y-coordinate
3	$\boldsymbol{z}(\mathbf{x}_c)$	z-coordinate
4	$\boldsymbol{t}(\mathbf{x}_c)$	time at arrival
5	$\boldsymbol{\chi}(\mathbf{x}_c)$	course at arrival
5	$\boldsymbol{\mu}(\mathbf{x}_c)$	bank angle at arrival
7	$\boldsymbol{G}(\mathbf{x}_c)$	actual start to candidate cost
8	$\boldsymbol{H}(\mathbf{x}_c)$	estimated candidate to goal cost
9	$\boldsymbol{F}(\mathbf{x}_c)$	estimated start to goal cost
10	$\boldsymbol{x}(\mathbf{x}_{in})$	x-coordinate of the parent state
11	$\boldsymbol{y}(\mathbf{x}_{in})$	y-coordinate of the parent state
12	$\boldsymbol{z}(\mathbf{x}_{in})$	z-coordinate of the parent state

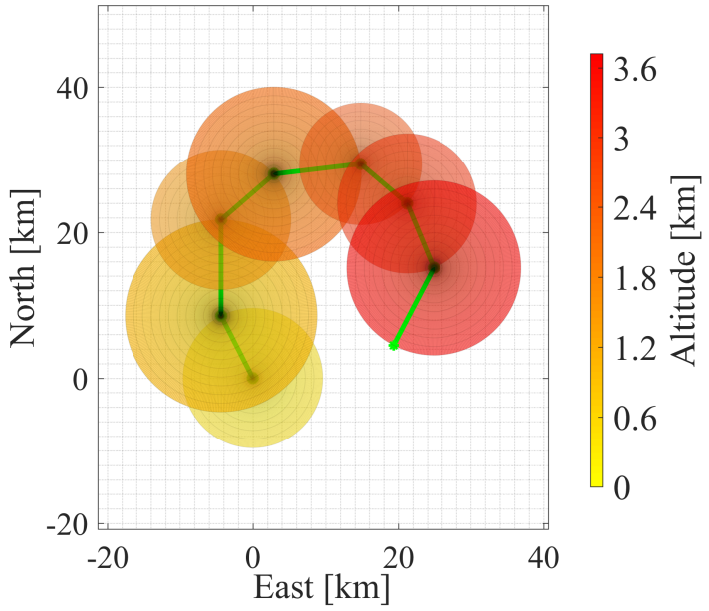
In each iteration, the most promising x_c, with the minimum estimated F-cost, is set as the next x_{in}. It is then popped from the priority queue (OL) and inserted into the so called closed list (CL), which stores already explored states. This process repeats, until either OL is empty, in which case the algorithm reports that no trajectory to the goal exists and performs partial planning (see ESA, Section 3.3), or the goal state x_g is inserted into CL, in which case the trajectory is constructed, by backtracking of the corresponding parent states and is then commanded to the flight controller.

Figure 4.15(a) depicts a five dimensional (east, north, altitude, course, time), discrete curvature-constrained and time-monotone trajectory, which is computed by solving the two dimensional problem in Figure 4.15(b). This is done, using the information stored in OL (Table 4.10) and under the assumption of constant speed $V_T = V_G$ and angle of climb γ. The resulting trajectory is a sequence of states, as shown in the example in Figure 4.14 (states connected by green arrows).

The following pseudocode (Algorithm 2), illustrates the flow of the MPTP, including the aforementioned A*-search. The user can choose between three heuristic functions and two options, regarding the handling of adjacent states x_a. The Option 1 in Algorithm 2, follows the procedure of the basic A*-search algorithm, as described in Section 4.2.1. Up to three queries are performed for each adjacent state x_a. The first query checks whether a state with the same position as x_a is already in CL. If not, the algorithm examines in a second query, whether a state, with the position as x_a, is already in OL. If not, x_a is



(**a**) *Three dimensional plot of a trajectory.*



(**b**) *Two dimensional view of a trajectory.*

Figure 4.15: *A dynamically feasible and time-monotone trajectory is computed in (a), by solving the two dimensional geometric problem in (b). The start state* x_s *on level* $t = 0$ *min in (a), is the first of seven initial states. The cones depicted in (a), are isochronous circles, which are necessary to create the estimated state space* EX *(see Section 3.3), from each of the seven initial states.*

inserted in OL. Otherwise, if a state with the same position as x_a, is already in OL, a third query examines whether the new G-cost is lower than before. If so, both the G-cost and parent state x_p are updated. In order to ensure the optimality of a trajectory, these

queries are necessary for environments containing static obstacles, since the estimated conflict area ECA of a static obstacle is the shape of the obstacle itself. Therefore, it can occur that positions of states are identical. Thus, it is recommended to use Option 1 for environments containing static obstacles. All results in Chapter 6 are conducted with Option 1 (OL- and CL-queries activated).

The first two queries have a worst case time complexity of $O(n)$, where n is the number of entries in the respective list. Especially OL can become very large, gradually slowing down the search. In environments, in which all obstacles are moving, the likelihood of repeating states is extremely low. This is due to the fact, that the shape of an ECA is a function of the future aircraft states from an initial state and an obstacle, as described in Section 3.3. Therefore, it is possible to leave out the aforementioned queries, by choosing Option 2. In environments containing exclusively moving obstacles, this can result in improved runtimes. However, this can be at the expense of optimality.

In Chapter 5, auxiliary states x_{aux} are introduced that are necessary to plan curvature-constrained trajectories and enable the planning of holding patterns (see Section 5.4). The position of auxiliary states can occur multiple times, for example in circular holding patterns, as depicted in Figure 5.10. These overlapping positions do only occur, if an auxiliary state was added as adjacent state x_a to another auxiliary state, which is then its parent state x_p. However, in this case the x_p is then inserted into CL. This makes it possible to plan holding patterns even if Option 1 is selected. However, this entails a limitation, in case that an inadmissible heuristic (see Section 4.3) is applied. In order to ensure the optimality of a trajectory, an additional query can be implemented in Algorithm 2, which examines if a state with the same position, as x_a, is already in CL. If the new G-cost is lower than before, both the cost and parent state x_p are updated, which hence could prevent the planning of automatic holding patterns.

If the position of the goal depends on time, the goal is a state x_g. Trajectory planning to x_g can be done in a reactive or tactical fashion. If no prediction for x_g exists, the actual x_g is updated by the mission planner in every MPTP iteration (Figure 2.3). The reactive guidance is likely to result in a trajectory with pursuit curve (see Figure 6.17(a), Section 6.4). If x_g can be predicted by the mission planner (see Figure 2.3), the goal state, with the minimal normal distance to the matching isochronous set of the aircraft, is selected for trajectory planning in every MPTP iteration (see Figure 6.17(b), Section 6.4). This results in the optimal trajectory to the goal state through the moving thunderstorms.

Algorithm 2: Model Predictive Trajectory Planning

Input: initial state $\mathrm{x}_{in}(x_{in}, y_{in}, z_{in}, \chi_{in}, t_{in})$ and latest thunderstorm nowcast
Output: anticipatory trajectory in free state space X_{free}

```
while aircraft has not arrived at the goal do
    load thunderstorms O_n(t) of the latest nowcast, add margins (see Section 3.1)
    perform spatiotemporal interpolation for the query times spaced by Δt
    update the actual aircraft state and set it as start state x_s
    if the goal state x_g will be uncovered sometime in the interval from t_0 to t_0 + T_N then
        initalize the open list OL = {x_in} with the start state x_s set as initial state x_in
        initalize empty closed list CL = ∅
        while OL ≠ ∅ do
            compute the set of estimated future states A_fut(x_in) for constant V_T
            compute the obstacle region EX_obs(x_in) = A_fut(x_in) ∩ O_n(t)
            compute the free state estimated space EX_free(x_in) = W \ EX_obs(x_in)
            generate the auxiliary state(s) x_aux (see upcoming Section 5)
            generate roadmap of free state space {V, E} = VG(EX_free(x_in))
            select adjacent states x_a to x_in, that meet dynamic constraints (Section 5)
            foreach adjacent state x_a do
                if Option 1: environment contains static and dynamic obstacles then
                    if x_a ∉ CL then
                        call one of three heuristic functions to compute H(x_a)
                        calculate total cost F(x_a) = G(x_a) + H(x_a)
                        if x_a ∉ OL then
                            insert x_a in OL
                        else
                            if G(x_a) < G(x_c) then
                                replace existing costs and set x_in as new parent state x_p
                            end
                        end
                    end
                else if Option 2: environment contains exclusively dynamic obstacles then
                    call one of three heuristic functions to compute H(x_a)
                    calculate total cost F(x_a) = G(x_a) + H(x_a) and insert x_a in OL
            end
            new x_in is x_c with minimum F-cost, remove x_c from OL and insert in CL
            if x_in is the goal state x_g then
                break
            end
        end
        if OL = ∅ then
            perform partial planning [47], e.g. flight inside estimated safe area ESA
        else
            construct the anticipatory trajectory by backtracking of the parent states x_p
            command trajectory states to the flight controller (see Figure 2.3)
        end
    else
        command straight flight in ESA or holding pattern (upcoming Section 5.4)
    end
end
```

4.3 Heuristics for the Informed Search

A heuristic is an valuable instrument to solve state-space problems [102]. For trajectory planning, the H-cost is the estimated time it takes, to get from an adjacent state x_a to the goal state x_g. In this thesis A*-search is used with three different heuristics, which are presented in the following. The choice of the heuristic function is decisive for the convergence of the search and the optimality of computed trajectories.

A heuristic is admissible, if it never (at any time or any partial solution) overestimates the costs-to-go. Furthermore, an admissible heuristic is consistent if the heuristic cost of the goal state is

$$H(\mathrm{x}_g) = 0 \tag{4.18}$$

and if it obeys the triangle inequality. Therefore,

$$H(\mathrm{x}_{in}) \leq C(\mathrm{x}_{in}, \mathrm{x}_a) + H(\mathrm{x}_a), \tag{4.19}$$

where C is the cost to get from x_{in} to x_a, has to be true for all existing states and their adjacent states. While a consistent heuristic is always admissible, an admissible heuristic can be inconsistent [102]. If the heuristic is at least admissible, A*-search is guaranteed to find the optimal solution, if one exists. If the H-cost is underestimated, the solution is optimal but unnecessary search is done, which slows down the convergence. Admissible heuristics cause A*-search to explore an exponential number of nodes [106] [107].

Inadmissible heuristics can overestimate the costs-to-go. However, they offer good performance in large search spaces, at the expense of optimality. This is why in some time-critical applications a deliberate overestimation of the heuristic costs is achieved using an inflation factor ($\sigma > 1$)

$$F(\mathrm{x}_a) = G(\mathrm{x}_a) + \sigma \cdot H(\mathrm{x}_a). \tag{4.20}$$

This provides A*-search a depth-first nature and speeds up the convergence, as fewer states are expanded. Bounds on the suboptimality can be given by this very inflation factor σ [106]. If the total cost is dominated by the heuristic cost

$$F(\mathrm{x}_a) = G(\mathrm{x}_a) + H(\mathrm{x}_a) \approx H(\mathrm{x}_a), \tag{4.21}$$

then A*-search turns into a greedy best-first search.

Optimally, the H-cost is exactly the real cost, in which case the A*-search only expands

the states of the optimal trajectory (see an example in Section 6.1, Figure 6.6(b)).

The time complexity of A*-search can be described by $O(b^d)$. The base b is the branching factor, which can be significantly influenced by the representation of the search (see Section 4.1.1). The exponent d is the depth of the search. While an admissible heuristic has no influence on the effective branching factor, the effective depth of the search can be reduced [108]. This is an important benefit, as the search space expands dynamically in the presented algorithm. A heuristic function can prevent an excessive growth of the searched state-space, as visible connections in EX_{free} are generated from each of the respective candidate states, with the lowest F-cost.

In the following Sections 4.3.1 to 4.3.3, three different heuristics for the A*-search are described. The selection of a heuristic function, in Algorithm 2, is made inside the for-loop, in line 18 for Option 1 and line 29 for Option 2.

4.3.1 Dijkstra's algorithm

The first and most ineffective search-method is Dijkstra's algorithm, which can be seen as special case of the A*-search algorithm, where the heuristic value is zero [109]. In this case, the total cost is calculated by the equation

$$F(\mathrm{x}_a) = G(\mathrm{x}_a) + 0. \tag{4.22}$$

The A*-search always expands the state with the minimum F-cost first. Therefore, in this case the state, which is nearest to the start state, is expanded first. The A*-search becomes Dijkstra's algorithm and the search expands, like a circular wavefront, in all directions. While the untargeted search is ineffective, it only requires a partial visibility graph VG_p (see Figure 4.1(c)), which contains the visible tangent edges of first order to the adjacent states and is considerably less expensive, than the computation of a complete VG_r. In each iteration of A*-search, a VG_p is computed from the initial state with the minimum F-cost (see line 32, in Algorithm 2). This process is repeated, until the goal state is reached, which results in partial shortest trajectory map (PSTM). This is analog to the partial shortest path map method, presented in Section 4.1.2. This incremental A*-search variant is guaranteed to find a near-optimal solution, if one exists, and is used to compare both the convergence and optimality of the following two A*-search variants, in Chapter 6.

4.3.2 Euclidean Distance Heuristic

The Euclidean distance heuristic (EDH) is frequently applied for A*-search. It is an admissible and consistent heuristic, as the costs-to-go are always underestimated and it obeys the triangle inequality, which guarantees, that the solution found is optimal [106]. In this case, it estimates the H-costs, to get from each x_a to x_g, by the beeline distance. As the computation of the Euclidean distance is inexpensive and A*-search with EDH only requires a VG_p (see Figure 4.1(c)), this variant is computationally very effective. As in Section 4.3.1, the incremental search results in a PSTM, which is guaranteed to contain the optimal solution, if one exists. However, EDH is not ideal for the task of avoiding time-varying thunderstorms, as obstacles are ignored by the Euclidean distance. Especially in dense scenarios, the negative effect becomes noticeable in extended runtimes (see Chapter 6).

4.3.3 Shortest Static Path Heuristic

This section features Contribution 5: *Heuristic for A*-Search in Dynamic Environments.*

Finally, a novel heuristic function for A*-search is introduced in this thesis. As mentioned in Section 4.2.2, the purpose of a VG_r is not only the determination of adjacent states x_a from an initial state x_{in}. It is furthermore used to calculate the shortest path from each x_a to the goal x_g. The length of a shortest path, which can be converted to time as V_T is constant, is applied as H-cost for the respective x_a. Therefore, the heuristic function is called shortest static path heuristic (SSPH). It uses a nested A*, which searches the actual VG_r of EX_{free} applying the Euclidean distance heuristic. Instead of determining the H-cost of each adjacent state in one reduced visibility graph VG_r, it can be determined alternatively by computing a partial shortest path map (see Section 4.1.2) from each adjacent state x_a to the goal state, by performing an A*-search as described in Section 4.2.1. Generally, the computation and memory requirements are considerably lower using this method.

In most cases the SSPH does underestimate the cost-to-go. This is due to the fact, that a shortest path is computed in the static estimated state space from an initial state. Therefore, every non-straight edge means a change in velocity, which can only be greater than the planning velocity. In the example in Figure 4.16, the length of $|s_1| < |s_2|$, if $\delta > 0°$. This implies that $V_1 = s_1/(t_2 - t_1) < V_2 = s_2/(t_2 - t_1)$. Thus, the time of arrival, which is the cost-to-go for the MPTP, is underestimated.

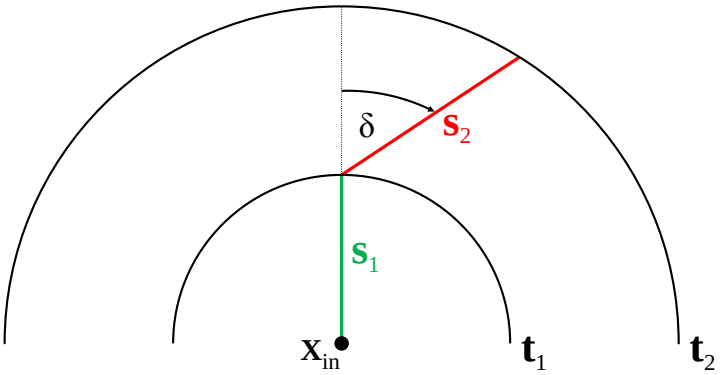


Figure 4.16: *The distance covered in $\Delta t = t_2 - t_1$, is greater than s_1, if the deviation from straight direction $\delta > 0°$. This error causes an underestimation of the H-cost by the SSPH, in the actual estimated state space, which is only valid from the initial state, from which it is computed.*

The obstacles (ECAs) for the static shortest path search are generated from an egocentric perspective from an initial state x_{in} (see Section 3.3). If a transition between subsequent isochronous circles is not in straight direction, this equals to a change in velocity, as more distance is covered in the same time interval Δt. Therefore, only the visible adjacent states x_a are valid. Nevertheless, the static graph of EX_{free} considers to some extent the obstacles, even if their shape is distorted. The dependency of the obstacle shape ECA and initial position is demonstrated in Figure 4.18. Depending on the relative movement between aircraft and obstacles, the ECA may appear to be larger than they actually are, which compensates the aforementioned underestimation and causes an overestimation of the costs-to-go. Since this cannot be ruled out, SSPH is a nonadmissible and inconsistent heuristic, which is why the optimality of trajectories cannot be guaranteed.

In [106] the quality of a heuristic function is characterized by the accuracy of its estimation, which is evaluated on the basis of an abstract analytical model. This is hardly feasible for SSPH. However, the accuracy can be determined experimentally. For this purpose, each x_a, for which a H-cost is computed, subsequently has to be the set as start state x_s, of a separate A*-search. An ensuing comparison between estimated and actual H-cost, allows a statistical assessment of the accuracy of the SSPH function, which however is only valid for the current configuration, i.e. ground speed V_G, maximum course change $\Delta\chi_{max}$ and time increment Δt.

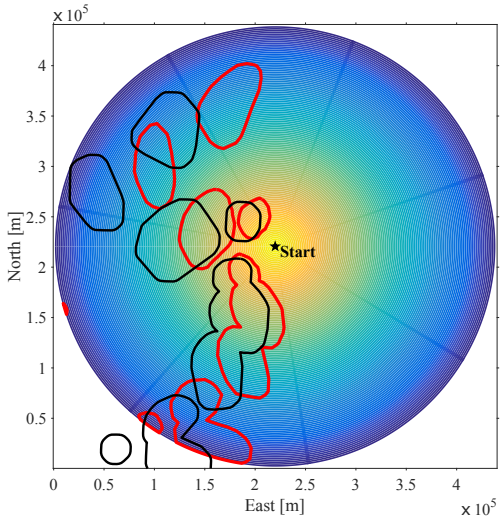
Nevertheless, the SSPH function can be characterized, as some parameters have a significant influence on its accuracy. The ratio R_{V_G}, between the ground speeds of the aircraft V_{G_A} and an obstacle V_{G_O}, has an important influence. This can be easily proven by the following thought experiment. If an aircraft travels at finite V_{G_A} and an obstacle is

static, the ground speed ratio is $R_{V_G} = V_{G_A}/0 = \infty$. In this case, the H-cost estimation by SSPH is exact, as the shapes of the ECAs and obstacle identical. As a consequence the A*-search exclusively expands the states of the optimal trajectory. Thus, it can be deduced, that R_{V_G} is decisive for the accuracy of SSPH. An example of this fact is given in Figure 4.17. Three aircraft traveling at ground speeds of $V_G = 60$ m/s (a), $V_G = 180$ m/s (b) and $V_G = 900$ m/s (c) are depicted. As expected, the fastest aircraft has the best match between, initial (thunderstorms at t_0) and estimated obstacles ECAs, as the ground speed ratio, between aircraft and thunderstorm, is the highest (Figure 4.17(c)). The SSPH function computes the H-cost based on the shortest distance to the goal state, in the visibility graph of the static free estimated state space EX_{free} (area outside the red polygons in Figure 4.17).

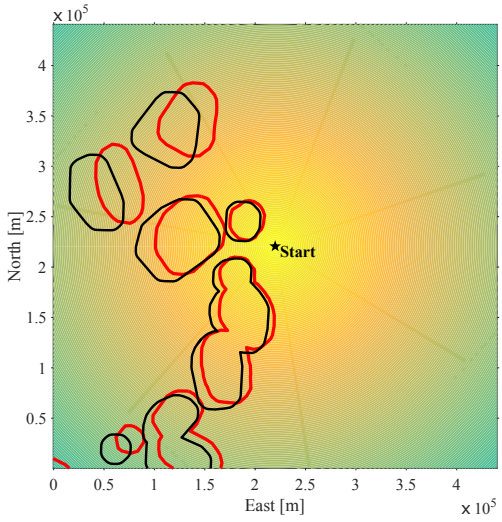
However, R_{V_G} is not the only parameter, that influences the accuracy of SSPH. Also, the relative position of x_{in}, with respect to the ground speed vector of an obstacle, is decisive for the shape of the resulting ECA. This directly influences the shortest path, which is the H-cost for the x_a. Figure 4.18 shows three examples, for ground speed ratios of 2.1, 2.9 and 20. In each Figure, the initial start state is localized in five different positions. Depending on ground speed ratio and relative placement (distance to the obstacle and lateral offset to the axis of motion) of the start state to the obstacle (red circle) axis of motion, shape and position of the respective ECA varies significantly. The shape of an ECA is generally closer to the real obstacle as R_{V_G} increases.

As mentioned before, a transition between subsequent isochronous circles, which is not in normal direction, is equivalent to an acceleration. Therefore, another factor for the accuracy of SSPH is the implicit change in velocity, if the shortest static path is not a straight line. The smaller the resulting sum of course changes of the shortest path inside the VG_r of EX_{free} is, the better the estimate by the SSPH function is. Therefore, it can be useful to calculate both the shortest path and the integrated velocity deviation (to the planning velocity) due to the course changes, in order to have a quality criterion for the prioritization of x_a. For this, however, the relationship between velocity deviation and accuracy of the H-cost estimate has to be determined. This can improve the accuracy of the SSPH function. Whether this is worth the additional computation has not been determined yet. Furthermore SSPH is only valid if a path does not return to previous isochrones, because the information on a previous isochrone is incorrect, as it is outdated.

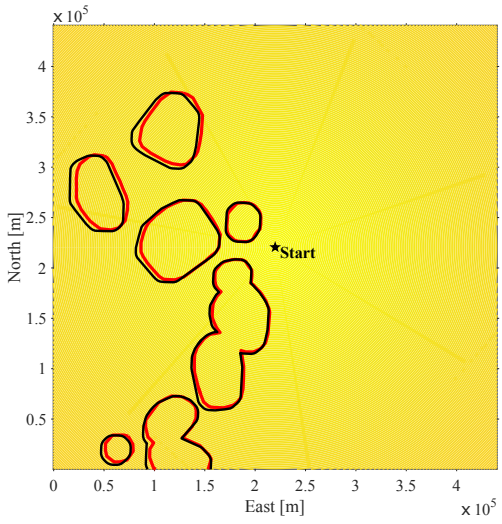
Although, the SSPH heuristic is computationally more expensive than the EDH function, the explicit consideration of obstacles speeds up the search in practice (see Chapter 6), as it reduces the expansion of the search space. This is especially valuable, if the



(**a**) *ECAs:* $V = 60$ *m/s and* $\Delta t = 30$ *s.*

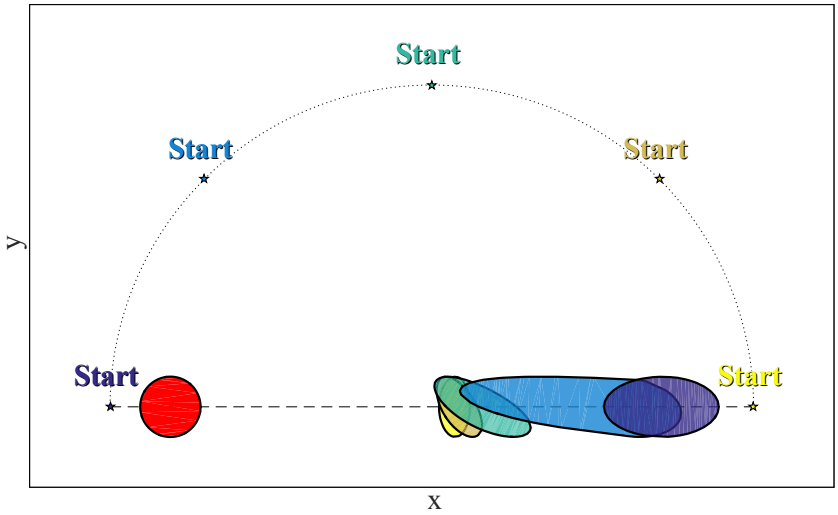


(**b**) *ECAs:* $V = 180$ *m/s and* $\Delta t = 10$ *s.*

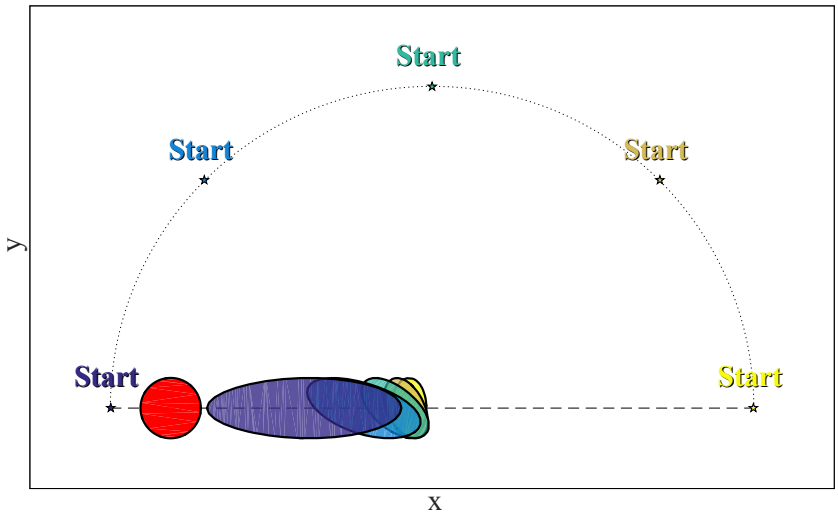


(**c**) *ECAs:* $V = 900$ *m/s and* $\Delta t = 2$ *s.*

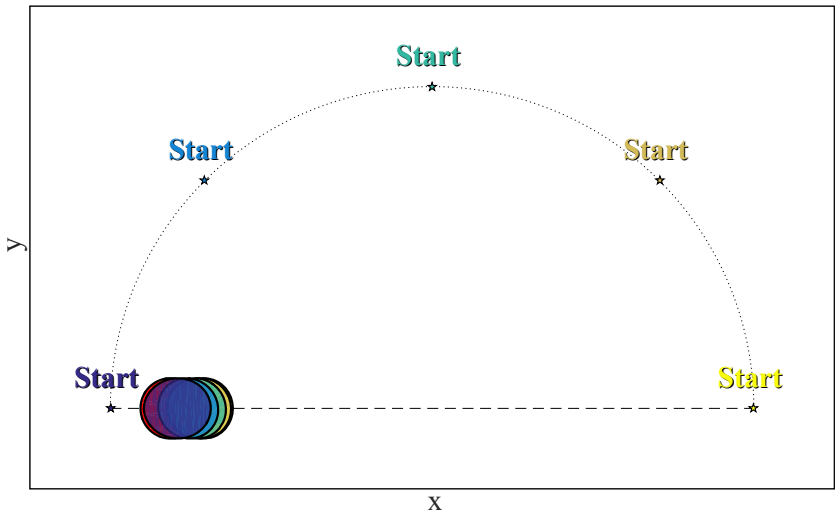
Figure 4.17: *The initial state is depicted by the black star in the center. The colormap indicates the positions of the aircraft, at times from* t_0 *(light yellow) to* $t_0 + 3600$ *s (dark blue). The respective* $ECA(\mathrm{x}_{in})$ *are depicted by red lines, while the nowcast measurement at* t_0 *(including safety margins) is depicted as black lines.*



(**a**) *ECAs for a ground speed ratio of 2.1.*



(**b**) *ECAs for a ground speed ratio of 2.9.*



(**c**) *ECAs for a ground speed ratio of 20.*

Figure 4.18: *In all figures an obstacle (red circle) translates in positive x-axis (right direction on the dashed line) at constant speed. From each start position (colored stars), placed along a half circle (dotted line), a unique ECA (color of associated start) of the moving obstacle (red circle) is computed.*

obstacles are large, as unnecessary search in impassable areas, can be prevented (see upcoming example in Figure 6.6). Alternatively to the presented nested A*-search approach, the fast marching method can be applied (see Section 3.2), to determine the static paths in $VG_r(\mathrm{x}_{in})$. In this case, no visibility graph is needed, which is especially interesting when searching three dimensional spaces.

Chapter 5

Modeling Nonholonomic Turning-Flight

Kinodynamic and nonholonomic/curvature-constrained motion planning are directly related [9]. A fixed-wing aircraft has velocity constraints in the y- and z-direction of its body-fixed frame of reference, as it cannot fly directly sideways or upwards. However, there are no restrictions in its reachable configurations, as they can be attained by performing a series of maneuvers. Therefore, a fixed-wing aircraft is a nonholonomic system [37]. If the future state primitives from Section 3.2 are used, the nonholonomic turning-flight constraint is considered innately. For the state sampling, which compute approximate future states, it has to be modeled explicitly. As the combinatorial planning approach from the previous chapter does not model the nonholonomic turning-flight by default, an auxiliary method is required to ensure that computed trajectories are dynamically feasible.

Dubins was one of the first to address the problem of finding the shortest curvature-constrained path, in the absence of obstacles [110]. When applied in a local planning method, it can be used to model nonholonomic constraints in the presence of both static and dynamic obstacles. For example using a RRT-planning approach [42], which uses three-dimensional Dubins path segments [111] for the tree-expansion [112]. This kind of forward simulation is typical for sampling-based motion planning approaches under differential constraints [42]. A system simulation module computes feasible state transitions from an initial state, by integration, e.g. using Runge-Kutta methods. If a simulated state does not already exist and not collide with an obstacle, it is added to a search graph [10]. A similar approach is applied in this thesis to consider nonholonomic constraints in combinatorial motion planning. The basic idea is to add flyable states, which in the following are called auxiliary states x_{aux}, to the list of adjacent states for each initial state x_{in}, in

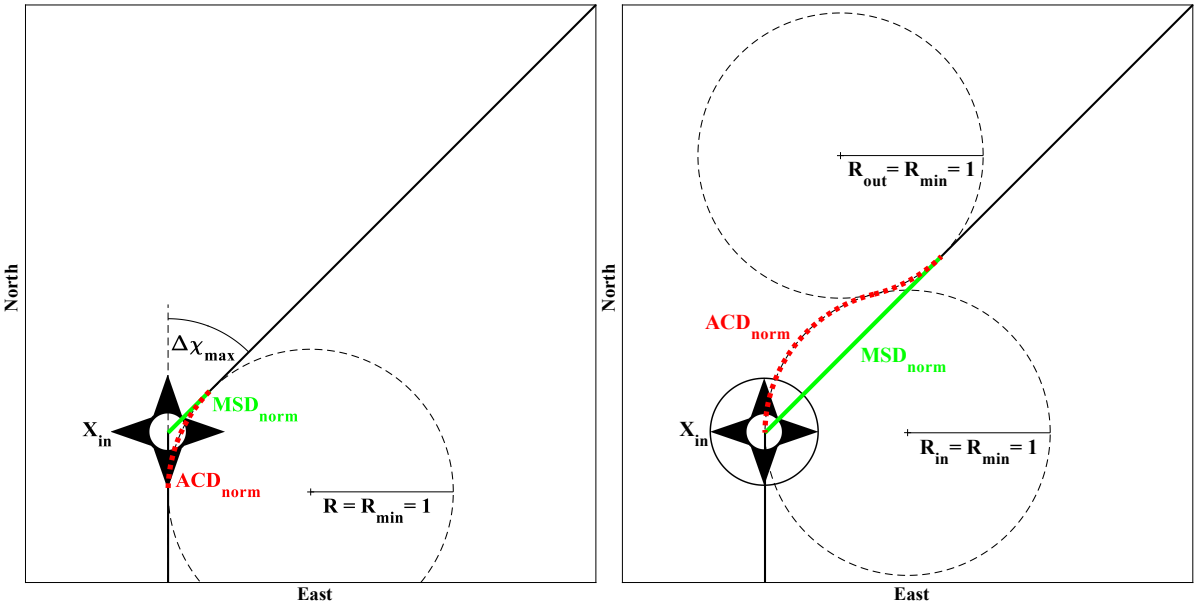
the A*-search (see Section 4.2.1), if neither they nor the trajectory to them are in conflict with obstacles. This results in discrete curvature-constrained time-minimal trajectories, which can be interpreted as a discrete version of Dubins path [113]. An auxiliary state can be generated for one turning-sense (unidirectional turning-flight) or both turning-senses (bidirectional turning-flight). In the following three sections, different methods for the computation of auxiliary states are presented, which range from low- to high-fidelity representation of the aircraft performance. Finally, a method for the automatic planning of holding patterns is introduced in Section 5.4, which enhances the capabilities of the MPTP considerably.

5.1 Simple Auxiliary States

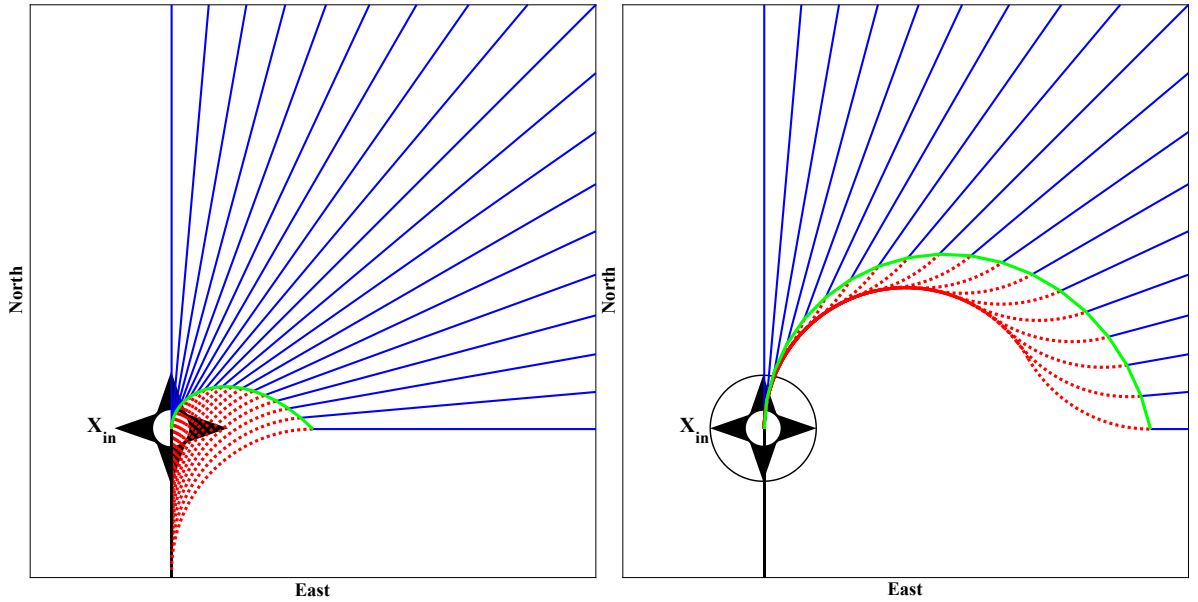
This section features part of Contribution 6: *Methods for the Computation of Feasible States.*

In this section, a straightforward method to compute future states or quickly evaluate their feasibility, even by hand, is introduced. Starting from x_{in}, the course change is limited to $|\Delta\chi_{max}| = 90°$. Adjacent future states to x_{in}, that require $|\Delta\chi| \geq |\Delta\chi_{max}|$ are disconnected from x_{in} in the VG. An x_{aux} is located on the new course at the leg distance of $LD = V_T \Delta t$ and can be performed as fly-by or fly-over state. In order to ensure, that a commanded state can be reached in due time, two conditions have to be met. First, the leg distance LD between the adjacent states has to be longer or equal than the minimum stabilization distance MSD [64]. The MSD is the minimum distance, that it takes for the aircraft to fly on the new course. Second, to reach the next commanded state (waypoint at a certain time) in due time, the aircraft has to fly with a corrected mean velocity V_{cor}. Generally, the distance covered by the aircraft (ACD) differs from the LD. However, the aircraft has to arrive at the next state in $t_{in} + \Delta t$. In case of a fly-by state and $0° \leq |\Delta\chi_{max}| \leq 90°$, the V_{cor} is $\leq V_T$. For a fly-over state, and $0° \leq |\Delta\chi_{max}| \leq 90°$, the true airspeed V_T is $\leq V_{cor}$. The V_{cor} has to be higher than the stall speed V_S and less or equal to maximum true airspeed $V_{T_{max}}$, to ensure feasibility of a trajectory.

In the following, a simple procedure to quickly assess the flyability of states, i.e. admissible combination of selected planning velocity V_T, Δt and $|\Delta\chi_{max}|$, is presented. With the setups in Figure 5.1(a) and (b), a normalized minimum stabilization distance MSD_{norm} and velocity correction δV are determined as a function of $|\Delta\chi_{max}|$ (see Tables 5.1 and 5.2), by setting the minimum turn radius to $R_{min} = 1$ (unit turning-circle). In order to determine the smallest possible $MSD = MSD_{norm} R_{min}$ for fly-over states, for



(**a**) *Geometric assumptions, for fly-by states.* (**b**) *Geometric assumptions, for fly-over states.*



(**c**) MSD_{norm} *for fly-by, as function of* $\Delta\chi_{max}$. (**d**) MSD_{norm} *for fly-over, as function of* $\Delta\chi_{max}$.

Figure 5.1: *Normalized aircraft covered distance and normalized minimum stabilization distance, when* x_{in} *is performed (a) as fly-by or (b) as fly-over state. An* x_{aux} *on the new leg has to be at least the minimum stabilization distance away from the* x_{in}*. The normalized minimum stabilization distance limit from* x_{in}*, for* $\Delta\chi_{max} = [0°, 90°]$*, is depicted by a green line in (c) for fly-by and in (d) for fly-over geometry. States that lie beyond the green* MSD_{norm} *line are feasible from* x_{in}*.*

both, the roll-in and roll-out radius, $R_{min} = 1$ is applied (see Figure 5.1(b)). Generally, roll-in and roll-out radius are different, leading to a smooth but longer MSD.

The values for the MSD_{norm} (Tables 5.1 and 5.2) are computed by the aforementioned approximation. Therefore, the establishment of the bank angle and the acceleration, for the correction of the distance per Δt error, are neglected. To account for these factors, a distance of $V_{cor}\delta t$ can be added to the MSD, for example with δt being 10 seconds, for a fly-over state [64]. Since the time increment Δt is a fixed value, the relative distance error between the normalized distance covered by the aircraft (ACD_{norm}) and the normalized minimum stabilization distance (MSD_{norm}), is used to calculate a velocity correction factor. Due to the turn anticipation, when performing a fly-by state, the planned distance corresponding to ACD_{norm} is twice the MSD_{norm} (see Figure 5.1(a)). Thus, the relative distance error for a fly-by state is calculated by

$$\epsilon_{rel} = \frac{ACD_{norm} - 2MSD_{norm}}{2MSD_{norm}} \tag{5.1}$$

and the relative distance error for a fly-over state is

$$\epsilon_{rel} = \frac{ACD_{norm} - MSD_{norm}}{MSD_{norm}}. \tag{5.2}$$

The velocity correction factor is calculated by

$$\delta V = 1 + \epsilon_{rel}. \tag{5.3}$$

Table 5.1: *Normalized minimum stabilization distance and velocity correction factors, as functions of $|\Delta\chi_{max}|$, for fly-by states.*

$\lvert\Delta\chi_{max}\rvert$	5	12	20	30	45	60	72
MSD_{norm}	0.044	0.105	0.176	0.268	0.414	0.577	0.727
δV	1.000	1.000	0.999	0.998	0.991	0.969	0.929

Table 5.2: *Normalized minimum stabilization distance and velocity correction factors, as functions of $|\Delta\chi_{max}|$, for fly-over states.*

$\lvert\Delta\chi_{max}\rvert$	5	12	20	30	45	60	72
MSD_{norm}	0.210	0.503	0.829	1.220	1.749	2.189	2.463
δV	1.001	1.005	1.015	1.033	1.078	1.139	1.206

The corrected mean velocity is then computed by

$$V_{cor} = V_T \delta V(|\Delta\chi_{max}|). \tag{5.4}$$

With the value for V_{cor} and $|\mu_{max}|$, the actual minimum turn radius can be computed

$$R_{min} = \frac{{V_{cor}}^2}{g \tan |\mu_{max}|}. \tag{5.5}$$

By multiplying $MSD_{norm}(|\Delta\chi_{max}|)$ with the minimum turn-radius $R_{min}(|\Delta\chi_{max}|)$, the actual minimum stabilization distance $MSD(|\Delta\chi_{max}|)$ is calculated. Table 5.3 shows exemplary values for different $|\mu_{max}|$.

Table 5.3: *Values for minimum stabilization distances for $V_T = 80$ m/s, $|\Delta\chi_{max}| = 45°$ (as used in Chapter 6) for different values of $|\mu_{max}|$, for fly-over and fly-by states.*

$\lvert\mu_{max}\rvert$ [°]	5	10	15	20	25	30	35	40	45
MSD_{ov} [m]	15091	7488	4928	3628	2831	2287	1886	1574	1320
MSD_{by} [m]	3033	1505	990	729	569	459	379	316	265

In order for a planned trajectory to be discretized in this way, the minimum time increment Δt is determined by

$$\Delta t_{min}(|\Delta\chi_{max}|, |\mu_{max}|) = \frac{MSD}{V_T}. \tag{5.6}$$

Notice, that the values for the velocity correction δV, in Tables 5.1 and 5.2, only apply for the special case, in which the leg distance LD is equal to the MSD. Otherwise, the corrected mean velocity V_{cor} is less, as the relative error between ACD and MSD is correspondingly smaller. This also implies that $R_{min}(V_{cor}, \mu_{max})$ is smaller. Therefore, if $MSD < LD$, an iterative calculation is applied, to solve for the necessary V_{cor}. For fly-over states this is described by

$$\delta V = 1 + \frac{ACD_{norm} R_{min}(V_T \delta V) - MSD_{norm} R_{min}(V_T \delta V)}{LD} = 1 + \frac{ACD - MSD}{LD}. \tag{5.7}$$

For the example, in Section 6.3, the leg distance is $LD = V_T \Delta t = 4800$ m. In the case of $LD = MSD$, $|\Delta\chi_{max}| = 45°$ and $\mu_{max} = 25°$, the corresponding value from Table 5.2 is $\delta V = 1.078$. Applying the aforementioned formula, with an convergence criterion of 10^{-9}, the new value is $\delta V = 1.042$. Therefore, the corrected mean velocity is $V_{cor} = V_T \delta V = 83.36$ m/s, instead of 86.24 m/s.

For values of $|\Delta\chi_{max}| \leq 90°$, fly-by states generally result in much shorter MSD and a less critical reduction of the velocity instead, compared to fly-over states. However, even if all states are fly-over, the simulations in Chapter 6 satisfy the criteria for flyability i.e. $MSD \leq LD = V_T \Delta t$ and $V_S < V_{cor} < V_{T_{max}}$.

In order to increase the robustness of the MPTP, $|\Delta\chi_{max}|$ can be defined as soft constraint. A x_{aux} is only valid, if it is neither in nor on $EX_{obs}(\mathrm{x}_{in})$. If no x_{aux} exists, $|\Delta\chi_{max}|$ can be increased stepwise, until a x_{aux} exists and $|\Delta\chi_{max}|$ is valid, i.e. the actual $MSD \leq LD$.

5.2 Approximate Auxiliary States

Alternatively, the auxiliary states can be computed for a $|\Delta\chi| \leq 90°$ and different values for $\mu \leq |\mu_{max}|$. The maximum bank angle $|\mu_{max}|$ is limited by the aircraft's dynamics and can be determined, using the relation between the lift and bank angle

$$mg = L_{max} \cos \mu_{max}, \tag{5.8}$$

which solved for the maximum bank angle yields

$$\mu_{max} = \arccos \frac{mg}{L_{max}}. \tag{5.9}$$

The maximum lift can be calculated by

$$L_{max} = C_{L_{max}} \bar{q} S, \tag{5.10}$$

where $\bar{q}$ is the dynamic pressure and S is the wing area of the aircraft.

In the following, a turn is approximated by a circular segment S_c, which neglects both increase and decrease of the bank angle and thus is only admissible for aircraft with a high roll rate p or for small bank angles. Due to the combination of circular and straight trajectory segments the approximate auxiliary state method is basically a Dubins path [110]. Smooth curvature transitions for aircraft with a slow roll rate can be described by clothoids [114, 115] or as presented later on, in Section 5.3, by using a Taylor series.

An x_{aux} is located at the end of a S_c, if the length

$$|S_c| = |\Delta\chi R| \tag{5.11}$$

is equal to $V_T\Delta t$ (see $x_{aux}(\mu = 4.6°)$ in Figure 5.2). The smallest possible bank angle μ_{min}, in order to meet this condition, while reaching the commanded course $\chi_{in} + \Delta\chi$, is calculated by

$$|\mu_{min}| = \arctan \frac{V_T \dot{\chi}}{g} \tag{5.12}$$

with

$$\dot{\chi} = \frac{\Delta\chi}{\Delta t}. \tag{5.13}$$

If $|S_c| < V_T\Delta t$, the x_{aux} is located at the end of a combination of S_c and a straight segment S_s (blue line, Figure 5.2), which is attached at the end of the circular segment S_c (red line in Figure 5.2) and whose length is

$$|S_s| = V_T\Delta t - |S_c|. \tag{5.14}$$

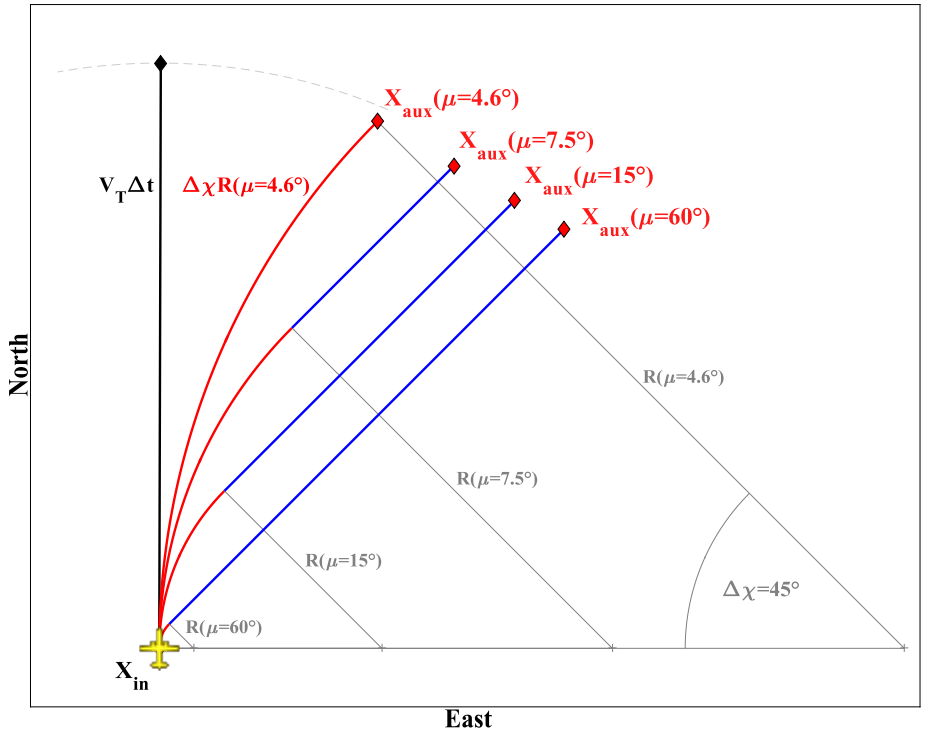


Figure 5.2: *Approximate auxiliary states, for constant $\Delta\chi$ and different values of μ.*

The Figure 5.2 shows an example for $\Delta\chi = 45°$ and different values of μ including $\mu_{min}(\Delta\chi = 45°) = 4.6°$, where the trajectory solely consist of a circular segment. By variation of μ the lateral displacement is increased. This method leads to more realistic x_{aux}, than those presented in Section 5.1, thereby improving the feasibility of the planned trajectory. In the next section, a forward simulation method is introduced.

5.3 Simulated Auxiliary States

This section features part of Contribution 6: *Methods for the Computation of Feasible States.*

The methods to generate auxiliary states, presented in Sections 5.1 and 5.2, are intended to minimize computational complexity and load. For aircraft with performance reserves and considering the large uncertainties in the forecast, it makes little sense to calculate elaborate x_{aux}. However, for low performance aircraft these methods may be insufficiently accurate, making it impossible for the flight controller to follow the commanded trajectory. This can result in large spatio-temporal errors. To ensure the feasibility of x_{aux}, an alternative method is introduced, which explicitly considers the dynamics of the aircraft (subscale to the planning time increment Δt of the MPTP). Instead of using a flight simulation, which is computationally expensive, as future states are computed using numerical integration of the aircraft's equations of motion, x_{aux} are computed using a Taylor series (TS), which is suitable for an online application.

In the following, the procedures to gather the necessary aircraft performance data are explained step by step. For demonstration purposes, a generic model of a HAPS (see Figure 5.3) is presented, whose design is inspired by the Zephyr 7 from Airbus.

5.3.1 Generic Low Performance Aircraft Model

A HAPS (high altitude pseudo-satellite) is a special class of unmanned aircraft designed for long term missions in the stratosphere. In order to achieve this, light weight construction (e.g. missing ailerons) and consequent energy saving (e.g. low thrust reserve) are crucial. Thus, a HAPS is ideally suited as an example for a low performance aircraft.

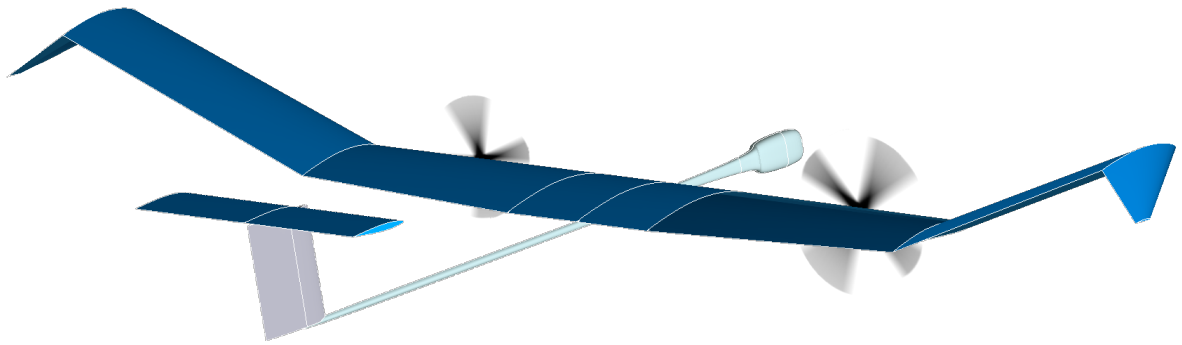


Figure 5.3: *An exemplary HAPS inspired by Airbus Zephyr 7 design with two engines.*

Table 5.4: *Specifications for the generic HAPS configuration.*

Aircraft Mass and Inertia		
$\boldsymbol{m}$	Curb Weight	100 [kg]
$\boldsymbol{I_{xx}}$	Rolling Moment of Inertia	1271.30 [kgm^2]
$\boldsymbol{I_{yy}}$	Pitching Moment of Inertia	238.50 [kgm^2]
$\boldsymbol{I_{zz}}$	Yawing Moment of Inertia	1488.30 [kgm^2]
$\boldsymbol{I_{xz}}$	Product of Inertia	-35.70 [kgm^2]
Propulsion		
$\boldsymbol{P_{max}}$	Engine Power	5000 [W]
$\boldsymbol{n_e}$	Number of Engines	4 [-]
$\boldsymbol{\eta_{pt}}$	Efficiency Powertrain	0.86 [-]
$\boldsymbol{\eta_{pr}}$	Efficiency Propeller	0.82 [-]
Main Wing		
-	Airfoil	Eppler $E395$
$\boldsymbol{b}$	Wingspan	22.50 [m]
$\boldsymbol{S}$	Wing Area	31.10 [m^2]
$\boldsymbol{\bar{c}}$	Mean Aerodynamic Chord	1.44 [m]
Horizontal Stabilizer		
-	Airfoil	NACA 63010a
-	Size	0.85×4.20 [m]
Vertical Stabilizer		
-	Airfoil	NACA 63010a
-	Size	1.50×1.00 [m]

5.3.2 Aerodynamic and Stability Analysis

In Tables 5.5 and 5.6, the dimensionless derivatives for the HAPS configuration are listed. All values are computed using the aerodynamic analysis software XFLR5 [116]. They are fixed and not a function of the Mach number Ma. Since the drag coefficient is obtained by a vortice lattice method (XFLR5) the viscous drag is not considered. Therefore, the polar is shifted towards a higher value of C_{D0} [117] (see Figure 5.4).

The aircraft design fulfills the criteria for longitudinal and lateral static stability. The damping derivatives C_{l_p}, C_{m_q} and C_{n_r} have negative sign. Furthermore C_{l_β} and C_{m_α} are also negative, while C_{n_β} is positive. As the center of gravity is before the neutral point, C_m is negative. At $\alpha(C_m = 0)$ the C_L is positive and (C_L/C_D) is close to $(C_L/C_D)_{max}$.

5.3.3 Estimation of True Air Speed Limits

The stall and maximum true air speeds at mean sea level (MSL) and an altitude of 19000 m are evaluated at international standard atmosphere (ISA) conditions. In the following, it is briefly presented how these values can be estimated, since they are necessary for the

Table 5.5: *Dimensionless longitudinal derivatives for the generic HAPS.*

Drag Force Coefficients	
C_{D_0}	0.014
C_{D_α}	0.100
C_{D_η}	0.008
C_{D_ζ}	0.010
Lift Force Coefficients	
C_{L_0}	0.380
C_{L_α}	5.550
C_{L_q}	7.482
C_{L_η}	0.487
Pitch Moment Coefficients	
C_{m_α}	-0.784
C_{m_q}	-15.931
C_{m_ξ}	0.005
C_{m_η}	-1.840
C_{m_ζ}	0.003

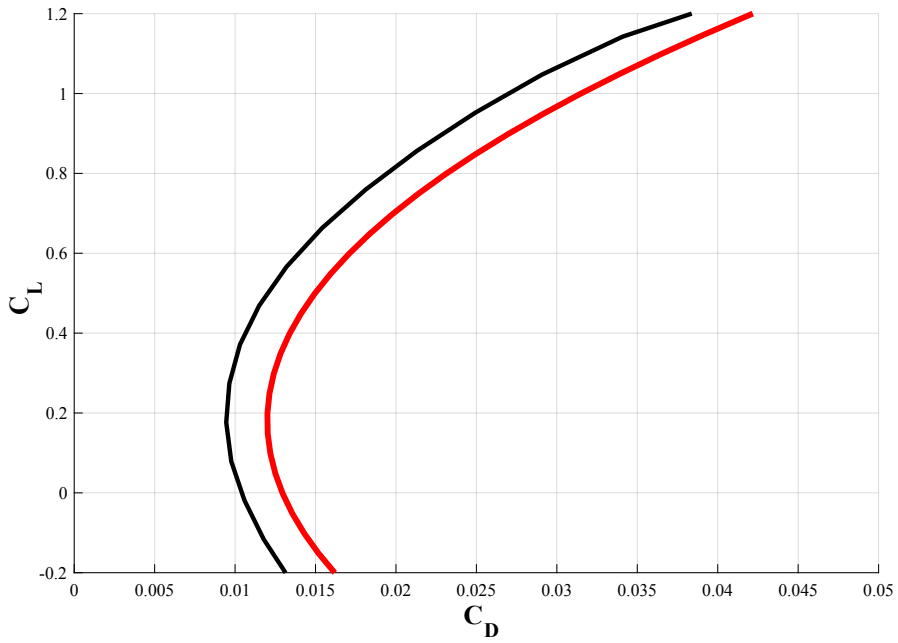


Figure 5.4: *As XFLR5 tends to underestimate the viscous drag, the C_L/C_D polar (red line) is shifted towards a higher zero lift drag with respect to the analysis (black line).*

forward simulation and trajectory planning.

The stall speed at MSL is approximately

$$V_{T_S}(h = 0m) = \sqrt{\frac{2mg}{\rho S C_{Lmax}}} = \sqrt{\frac{2 \cdot 100 \text{ kg} \cdot 9.81 \text{m/s}^2}{1.225 \text{ kg/m}^3 \cdot 31.1 \text{ m}^2 \cdot 1.35}} = 6.18 \text{ m/s}. \tag{5.15}$$

Table 5.6: *Dimensionless lateral derivatives for the generic HAPS.*

Side Force Coefficients	
$\boldsymbol{C_{Y_\beta}}$	-0.266
$\boldsymbol{C_{Y_p}}$	-0.275
$\boldsymbol{C_{Y_r}}$	0.119
$\boldsymbol{C_{Y_\xi}}$	0.277
$\boldsymbol{C_{Y_\zeta}}$	0.080
Yaw Moment Coefficients	
$\boldsymbol{C_{n_\beta}}$	0.013
$\boldsymbol{C_{n_p}}$	-0.065
$\boldsymbol{C_{n_r}}$	-0.014
$\boldsymbol{C_{n_\zeta}}$	-0.021
Roll Moment Coefficients	
$\boldsymbol{C_{l_\beta}}$	-0.163
$\boldsymbol{C_{l_p}}$	-0.589
$\boldsymbol{C_{l_r}}$	0.117
$\boldsymbol{C_{l_\xi}}$	-0.109
$\boldsymbol{C_{l_\zeta}}$	0.003

According to WGS84 the gravitational acceleration at an altitude of $h = 19000$ m is

$$g(h) = \frac{GM}{(R+h)^2} = \frac{6.674 \cdot 10^{-11}\text{m}^3/\text{kgs}^2 \cdot 5.972 \cdot 10^{24}\ \text{kg}}{(6371000\ \text{m} + 19000\ \text{m})^2} = 9.76\ \text{m/s}^2, \tag{5.16}$$

with G being the gravitational constant, M the mass of the earth and R its mean radius. The stall true air speed at altitude can be estimated by

$$V_{T_S}(h) = \sqrt{\frac{2mg}{\rho S C_{Lmax}}} = \sqrt{\frac{2 \cdot 100\ \text{kg} \cdot 9.76\ \text{m/s}^2}{0.121\ \text{kg/m}^3 \cdot 31.1\ \text{m}^2 \cdot 1.35}} = 19.60\ \text{m/s}. \tag{5.17}$$

For steady level flight the power can be written as thrust times velocity

$$P = TV_T. \tag{5.18}$$

In steady level flight the equilibrium condition is, that thrust equals drag

$$T = D. \tag{5.19}$$

Using Equation (5.19) the Equation (5.18) can be written as

$$P = DV_T, \tag{5.20}$$

where the drag is

$$D = C_D \bar{q} S. \tag{5.21}$$

The required lift for steady level flight is

$$L = mg = C_L \bar{q} S, \tag{5.22}$$

which solved for the lift coefficient for level flight is

$$C_{L_{lev}} = \frac{mg}{\bar{q}S} = \frac{2mg}{\rho S V_T^2}. \tag{5.23}$$

The drag coefficient for level flight is described by

$$C_{D_{lev}} = C_{D_{min}} + k_1 (C_{L_{lev}} - k_2)^2 = 0.012 + 0.029 (C_{L_{lev}} - 0.18)^2, \tag{5.24}$$

where k_1 is a constant and $k_2 = C_L(C_{D_{min}})$ (see Figure 5.4). Using Equations (5.21), (5.23) and (5.24) Equation (5.20) can be written as

$$n_e P_{max} \eta_{to} = D V_{T_{max}} = \bar{q} S C_{D_{lev}} V_{T_{max}} = \frac{\rho S}{2} \left(C_{D_{min}} + k_1 \left(k_2 - \frac{2mg}{\rho S V_{T_{max}}^2} \right)^2 \right) V_{T_{max}}^3, \tag{5.25}$$

where n_e is the number of engines and η_{to} is the total efficiency of powertrain η_{pt} and propeller η_{pr}

$$\eta_{to} = \eta_{pt} \cdot \eta_{pr} = 0.86 \cdot 0.82 = 0.7. \tag{5.26}$$

Solved for the maximum true airspeed this yields $V_{T_{max}}(h = 0 \text{ m}) = 38.76$ m/s, which corresponds to $Ma = 0.11$, and $V_{T_{max}}(h = 19000 \text{ m}) = 89.21$ m/s, which corresponds to $Ma = 0.30$.

5.3.4 Nonlinear 6-DoF Flight Simulation

The flight simulation, which is described in this section, is implemented in Simulink. Most of the equations in this section are given by [118].

Assumptions for the Simulation

- Flat nonrotating earth.
- No relative velocities in the body frame. Aeroelastic effects are not modelled, although a HAPS is a very flexible configuration. The stiff model is sufficiently accu-

rate to evaluate the average performance of the configuration.

- The air is incompressible, as the aircraft flies in the Mach range Ma ≤ 0.3.

Frames of Reference

Body Frame of Reference:
The body frame of reference (index B) is used for the calculation of forces and moments. It's origin is at the center of gravity (CG) and translates and rotates with the aircraft. The positive x_B-axis points toward the nose, the y_B-axis towards the right wing and the z_B-axis towards the bottom of the aircraft. The velocity in x_B-direction is denoted by u, in y_B-direction by v and in z_B-direction by w. The roll rate around x_B is denoted by p, the pitching rate around y_B is q and the yaw rate around z_B is r.
Aerodynamic Frame of Reference:
Origin of the aerodynamic axis system (index A) is at the CG of the aircraft. First, the B-frame is rotated around the y_B-axis, by the angle of attack $-\alpha$, and then around its z_A-axis, by the sideslip angle β. The x-axis of the A-frame is aligned with the velocity of the free airflow V

$$|V| = \sqrt{u^2 + v^2 + w^2}, \tag{5.27}$$

with

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} V \cos\alpha \cos\beta \\ V \sin\beta \\ V \sin\alpha \cos\beta \end{bmatrix}. \tag{5.28}$$

The aerodynamic angles can be expressed in B-frame velocities. The angle of attack is

$$\alpha = \arctan \frac{w}{u} \tag{5.29}$$

and the sideslip angle is

$$\beta = \arcsin \frac{v}{V}. \tag{5.30}$$

The derivatives of the aerodynamic angles can be calculated by the formulas given in [118]

$$\dot{\alpha} = \frac{\dot{w}\cos\alpha - \dot{u}\sin\alpha}{V\cos\beta}, \tag{5.31}$$

$$\dot{\beta} = \frac{-\dot{u}\cos\alpha\sin\beta + \dot{v}\cos\beta - \dot{w}\sin\alpha\sin\beta}{V}. \tag{5.32}$$

Inertial Earth Fixed Frame of Reference:
The earth fixed frame of reference (index E) serves to determine the position of the aircraft. The earth is assumed to be flat and does not rotate. This is sufficiently accurate for short term simulations. The x_E-axis points towards geographic north and the y_E-axis towards geographic east. Both are parallel to the local surface of the geoid. The z_E-axis is perpendicular to plane definded by x_E and y_E and points towards the center of the earth. The transformation matrix M, from E- to the B-frame of reference, is given by

$$\mathbf{M}_{BE} = \begin{bmatrix} \cos\Psi\cos\Theta & \sin\Psi\cos\Theta & -\sin\Theta \\ \cos\Psi\sin\Theta\sin\Phi - \sin\Psi\cos\Phi & \sin\Psi\sin\Theta\sin\Phi + \cos\Psi\cos\Phi & \cos\Theta\sin\Phi \\ \cos\Psi\sin\Theta\cos\Phi + \sin\Psi\sin\Phi & \sin\Psi\sin\Theta\cos\Phi - \cos\Psi\sin\Phi & \cos\Theta\cos\Phi \end{bmatrix} \tag{5.33}$$

where Ψ is the azimuth angle, Θ is the pitch angle and Φ is the bank angle. Thus, the gravitational acceleration can be transformed from E- to the B-frame by

$$\begin{bmatrix} g_{x_B} \\ g_{y_B} \\ g_{z_B} \end{bmatrix} = \mathbf{M}_{BE} \begin{bmatrix} g_{x_E} \\ g_{y_E} \\ g_{z_E} \end{bmatrix}. \tag{5.34}$$

Definition of State and Control Vector

The state vector is defined as

$$\begin{bmatrix} u \\ v \\ w \\ p \\ q \\ r \\ x \\ y \\ z \\ \Phi \\ \Theta \\ \Psi \end{bmatrix} = \begin{bmatrix} \text{velocity in x-axis of the body frame, in [m/s]} \\ \text{velocity in y-axis of the body frame, in [m/s]} \\ \text{velocity in z-axis of the body frame, in [m/s]} \\ \text{roll rate around x-axis of the body frame, in [rad/s]} \\ \text{pitch rate around y-axis of the body frame, in [rad/s]} \\ \text{yaw rate around z-axis of the body frame, in [rad/s]} \\ \text{center of gravity position in direction north, in [m]} \\ \text{center of gravity position in direction east, in [m]} \\ \text{center of gravity position in direction down, in [m]} \\ \text{bank angle of the aircraft w.r.t earth frame, in [rad]} \\ \text{pitch angle of the aircraft w.r.t earth frame, in [rad]} \\ \text{azimuth angle of the aircraft w.r.t earth frame, in [rad]} \end{bmatrix}. \tag{5.35}$$

The control vector is defined as

$$\begin{bmatrix} \delta\xi \\ \delta\eta \\ \delta\zeta \\ \delta_T \end{bmatrix} = \begin{bmatrix} \text{aileron deflection, in [rad]} \\ \text{elevator deflection, in [rad]} \\ \text{rudder deflection, in [rad]} \\ \text{throttle, in [\%]} \end{bmatrix}. \tag{5.36}$$

Total Coefficients

The total lift-force coefficient is calculated by

$$C_L = C_{L_0} + C_{L_a}\alpha + C_{L_{\dot{\alpha}}}\dot{\alpha}\frac{\bar{c}}{2V} + C_{L_q}q\frac{\bar{c}}{2V} + C_{L_\eta}\delta\eta. \tag{5.37}$$

According to Equation (5.24) the total drag force-coefficient is

$$C_D = C_{D_{min}} + k_1(C_L - k_2)^2. \tag{5.38}$$

The total pitching-moment coefficient is

$$C_m = C_{m_0} + C_{m_\alpha}\alpha + C_{m_q}q\frac{\bar{c}}{2V} + C_{m_\xi}\delta\xi + C_{m_\eta}\delta\eta + C_{m_\zeta}\delta\zeta. \tag{5.39}$$

The total side-force coefficient is

$$C_Y = C_{Y_\beta}\beta + C_{Y_p}p\frac{b}{2V} + C_{Y_r}r\frac{b}{2V} + C_{Y_\xi}\delta\xi + C_{Y_\zeta}\delta\zeta. \tag{5.40}$$

The total yawing-moment coefficient is

$$C_n = C_{n_\beta}\beta + C_{n_p}p\frac{b}{2V} + C_{n_r}r\frac{b}{2V} + C_{n_\zeta}\delta\zeta. \tag{5.41}$$

The total rolling-moment coefficient is

$$C_l = C_{l_\beta}\beta + C_{l_p}p\frac{b}{2V} + C_{l_r}r\frac{b}{2V} + C_{l_\xi}\delta\xi + C_{l_\zeta}\delta\zeta. \tag{5.42}$$

The force coefficients in the x_B- and z_B-axis are computed by

$$C_X = -C_D\cos\alpha\cos\beta + C_L\sin\alpha, \tag{5.43}$$

$$C_Z = -C_D\sin\alpha\cos\beta - C_L\cos\alpha. \tag{5.44}$$

Forces and Moments

The actual forces and moments acting on the aircraft are necessary to calculate the accelerations in the B-frame. For this purpose the dynamic pressure is required, which is calculated by

$$\bar{q} = \frac{1}{2}\rho V^2. \tag{5.45}$$

With the total coefficients from the previous section, the forces, acting on the aircraft in body axis reference frame, are calculated by

$$X = C_X \bar{q} S + T_B, \tag{5.46a}$$

$$Y = C_Y \bar{q} S, \tag{5.46b}$$

$$Z = C_Z \bar{q} S, \tag{5.46c}$$

where T_B is the thrust, which is assumed to act in x_B-direction and to cause no additional moments. The moments in body axis are calculated by

$$L = C_l \bar{q} S b, \tag{5.47a}$$

$$M = C_m \bar{q} S \bar{c}, \tag{5.47b}$$

$$N = C_n \bar{q} S b, \tag{5.47c}$$

where b is the wingspan and $\bar{c}$ is the mean aerodynamic chord. The load factors in body axis can be computed by

$$n_x = \frac{X}{mg}, \tag{5.48a}$$

$$n_y = \frac{Y}{mg}, \tag{5.48b}$$

$$n_z = \frac{-Z}{mg}. \tag{5.48c}$$

Equations of Motion

The translational accelerations are calculated by

$$\dot{u} = \frac{X}{m} + g_{x_B} + rv - qw, \tag{5.49a}$$

$$\dot{v} = \frac{Y}{m} + g_{y_B} - ru + pw, \tag{5.49b}$$

$$\dot{w} = \frac{Z}{m} + g_{z_B} + qu - pv. \tag{5.49c}$$

The angular accelerations are calculated by

$$\dot{p} = \frac{I_{zz}L + I_{xz}N - (I_{xz}(I_{yy} - I_{xx} - I_{zz})p + (I_{xz}^2 + I_{zz}(I_{zz} - I_{yy}))r)q}{I_{xx}I_{zz} - I_{xz}^2}, \tag{5.50a}$$

$$\dot{q} = \frac{M - (I_{xx} - I_{zz})pr - I_{xz}(p^2 - r^2)}{I_{yy}}, \tag{5.50b}$$

$$\dot{r} = \frac{I_{xz}L + I_{xx}N + (I_{xz}(I_{yy} - I_{xx} - I_{zz})r + (I_{xz}^2 + I_{xx}(I_{xx} - I_{yy}))p)q}{I_{xx}I_{zz} - I_{xz}^2}. \tag{5.50c}$$

If wind is considered the velocities of the aircraft's CG in the E-frame (navigation equations) are

$$\begin{bmatrix} \dot{x}_E \\ \dot{y}_E \\ \dot{z}_E \end{bmatrix} = \mathbf{M}_{EB} \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} W_x \\ W_y \\ W_z \end{bmatrix}, \tag{5.51}$$

where ${\mathbf{M}_{BE}}^T = \mathbf{M}_{EB}$ and W is the wind speed vector relative to E-frame.

The kinematic equations, using Euler angles, are

$$\dot{\Phi} = p + (\sin\Phi q + \cos\Phi r)\tan\Theta, \tag{5.52a}$$

$$\dot{\Theta} = \cos\Phi q - \sin\Phi r, \tag{5.52b}$$

$$\dot{\Psi} = \frac{(\sin\Phi q + \cos\Phi r)}{\cos\Theta}. \tag{5.52c}$$

At the beginning of a simulation the integration blocks are initialized with start values. The state derivatives are integrated and fed back as the new actual states.

5.3.5 Assessment of Steady-State Angular Rates

Tight turns or rather small turning-radii can cause a strongly asymmetric lift distribution. If the lateral stability cannot compensate the differential lift, the result is an overbanking tendency (OBT), which in aircraft like HAPS can already occur at small bank angles. This effect is not reflected by the equations of motion in Section 5.3.4. If the aircraft overbanks it is necessary to deflect the aileron in the opposite direction in order to avoid an increasing bank, which as a consequence reduces the available deflection for the roll-out. As a result the roll-in rate p_{ri} will be higher than the roll-out rate p_{ro} for the same deflection. To exclude effects due to OBT, the bank angle is limited to $\mu_{max} = 15°$ and the minimum true airspeed to $V_T = 22$ m/s. Thus, the resulting minimum turn radius is

$$R_{min} = \frac{V^2}{g\tan\mu_{max}} = \frac{(22\text{ m/s})^2}{9.81\text{ m/s}^2\tan(15°)} = 184.13\,m. \tag{5.53}$$

Using the formula

$$V = \omega R_{min} \tag{5.54}$$

the angular velocity can be calculated, which in this case is $\omega = 0.119$ rad/s along the wing span. Considering the bank angle of $\mu_{max} = 15°$, the inner and outer wingtip are at a distance of $d = \pm\cos(\mu_{max})b/2 = \pm 10.87$ m to the center line of the aircraft. With an airflow velocity of approximately 20.70 m/s at the inner and 23.30 m/s at the outer wingtip the resulting differential lift is not estimated to be significant. Furthermore, the pronounced dihedral of HAPS (see Figure 5.3) has a counteractive effect. For this reasons OBT will not be considered further from here on. Additionally, for the considerations in the following Section 5.3.6, it is assumed that the roll-in rate p_{ri} and roll-out rate p_{ro} are approximately equal, provided that an appropriate controller is equipped and sufficiently aileron deflection reserve is available. An aileron step input causes a PT_1 roll response, for which a time constant τ can be estimated by

$$\tau = -\frac{1}{L_p} = \frac{-I_{xx}}{\frac{\rho}{2}SV\left(\frac{b}{2}\right)^2 C_{l_p}}, \tag{5.55}$$

according to [119]. At an altitude of $h = 10000$ m and $V = 22$ m/s, for example, this yields an approximate roll time constant for the HAPS of $\tau = 0.12$ s. At $t = 5\tau = 0.6$ s the roll rate is near its steady-state value

$$p(t) = p_{ss}\left(1 - \mathrm{e}^{-t/\tau}\right) = p_{ss}(1 - \mathrm{e}^{-5}) = 0.993 p_{ss}. \tag{5.56}$$

With the previously introduced simulation model, it is possible to assess reasonably accurate values for the roll rate, by trimming steady-state flight conditions, using the linear analysis toolbox in Simulink. Table 5.7 shows results for different altitudes and velocities. Tables 5.8 and 5.9 contain the corresponding trim specifications regarding the inputs and states.

Table 5.7: *Steady-state roll rate for an aileron deflection of $\delta\xi = -1°$ at altitudes of $h = 1000$ m and $h = 10000$ m.*

V_T[m/s]	22	26	30	34	38
$p_{ss}(h = 1000\,m)$ $[°/s]$	0.342	0.404	0.466	0.529	0.591
$p_{ss}(h = 10000\,m)[°/s]$	0.208	0.246	0.284	0.322	0.360

Table 5.8: *State specifications to obtain the steady-state rates in Table 5.7.*

State	Value	Known	Steady-State
$\boldsymbol{u}$	V_T	×	×
$\boldsymbol{v}$	0 m/s	×	×
$\boldsymbol{w}$	-	-	×
$\boldsymbol{p}$	-	-	×
$\boldsymbol{q}$	0 °/s	×	×
$\boldsymbol{r}$	0 °/s	×	×
$\boldsymbol{x}$	-	-	-
$\boldsymbol{y}$	-	-	-
$\boldsymbol{z}$	h	×	×
$\boldsymbol{\Phi}$	-	-	-
$\boldsymbol{\Theta}$	-	-	×
$\boldsymbol{\Psi}$	-	-	-

Table 5.9: *Input specifications to obtain the steady-state rates in Table 5.7.*

State	Value	Known
$\boldsymbol{\delta\xi}$	$-1°$	×
$\boldsymbol{\delta\eta}$	-	-
$\boldsymbol{\delta\zeta}$	-	-
$\boldsymbol{\delta_T}$	-	-

5.3.6 Auxiliary States Computation using Taylor Series

The objective of this section is to determine dynamically feasible auxiliary states x_{aux}, which can be achieved exactly at the planner's time increment Δt on a commanded course. For this purpose, a fast and reasonably accurate computation method is introduced that consists of a simplified dynamic aircraft model and a noniterative simulation method, i.e. the Taylor series (TS). Compared to iterative methods, for example Runge-Kutta (RK), it has the disadvantage that the integration cannot be stopped at certain events, for example if the maximum bank angle μ_{max} or desired course change $\Delta\chi$ is achieved. Therefore, the simulation times at which these events occur have to be precisely specified in advance. For the present application, this can be accomplished analytically, assuming that roll-in and roll-out rate are approximately equal and piecewise constant. The presented method is suitable for real time application and accurate within the limits of the simplifications made, if the chosen integration interval is not too large or the TS order is high enough.

The Taylor series is based on the Taylor theorem

$$f(t) = f(t_{in}) + \sum_{n=1}^{\infty} \frac{f^{(n)}(t_{in})}{n!}(t - t_{in})^n, \tag{5.57}$$

where $f(t_{in})$ is the initial state. For the implementation of TS integration, automatic differentiation is applied, as described in [120]. This further speeds up the running times, as the number of terms to evaluate is reduced. The Matlab implementation is from [121], where a detailed description can be found. TS uses the following differential equations for planar motion

$$u = V \cos \chi, \tag{5.58a}$$

$$v = V \sin \chi, \tag{5.58b}$$

$$\dot{\chi} = \frac{g \tan \mu}{V}, \tag{5.58c}$$

$$\dot{\mu} = p_{ss}. \tag{5.58d}$$

A turn from an initial state x_{in}, as center of a Taylor series (TS), to an auxiliary state x_{aux} always begins with $p_{in} = 0\ ^\circ/s$, $\mu_{in} = 0^\circ$, $\Delta\chi_{in} = 0\ ^\circ$ and ends with $p_{fin} = 0\ ^\circ/s$, $\mu_{fin} = 0^\circ$ on the desired course. Generally, initial conditions are taken from the navigation graph (see Table 4.10, Section 4.2.2), which stores the information about candidate states x_{in}. The first segment, which is always the roll-in, is initialized with these values and each following one with the last state of the previous segment.

Figure 5.5 shows four cases for turning-flight, which consist of at least two segments (roll-in and roll-out) and a maximum of four segments (roll-in, constant bank, roll-out, straight). The temporal limit for the cases in Figure 5.5(a) and (b) is given by

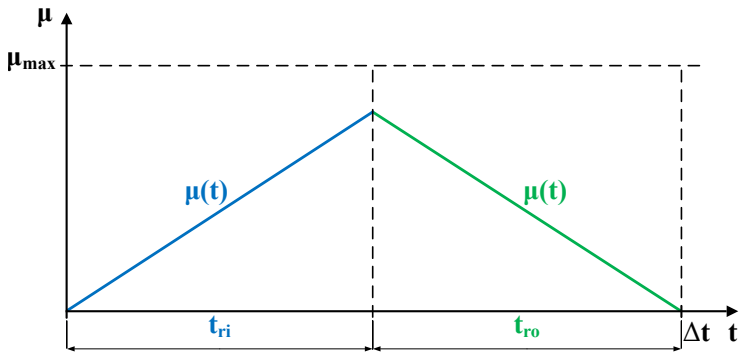
$$t_{\mu_{max}} = \frac{\mu_{max}}{p_{ss}}. \tag{5.59}$$

Under the assumption $p_{ss} = p_{ri} = p_{ro}$ and infinite roll rate acceleration, the necessary half roll time t_r, to achieve a desired course change, can be established using the integral of Equation (5.58c) in the form

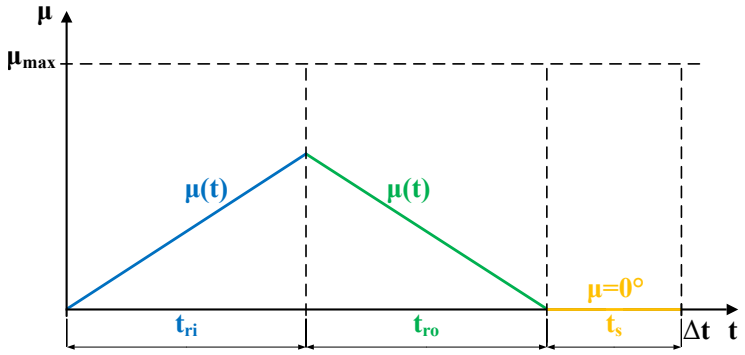
$$\Delta\chi = \frac{g}{V}\left(\int_0^{t_r} \tan(\mu_{ri}(t))dt + \int_0^{t_r} \tan(\mu_{ro}(t))dt\right), \tag{5.60}$$

which using a computer algebra system (CAS) results in

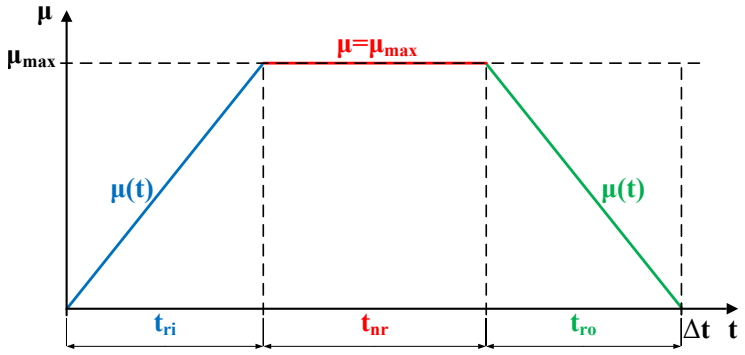
$$\Delta\chi = \frac{g}{V}\left(-\frac{\ln(\cos(p_{ss}t_r))}{p_{ss}} + \frac{\ln(\cos(\mu_{ri} - p_{ss}t_r)\sec(\mu_{ri})}{p_{ss}}\right), \tag{5.61}$$



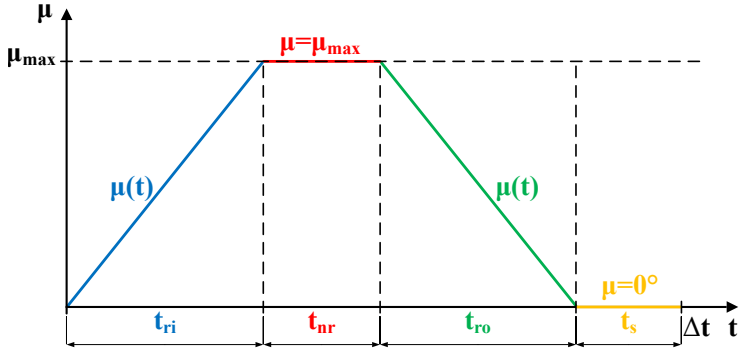
(**a**) *Limit case: Turn consisting of roll-in and roll-out segment.*



(**b**) *Turn consisting of roll-in, roll-out and straight segment.*



(**c**) *Limit case: Turn consisting of roll-in, no-roll and roll-out.*



(**d**) *Turn consisting of four consecutive segments.*

Figure 5.5: *Turn combinations for the construction of auxiliary states with TS integration. The roll-in segment is depicted in blue, the constant bank angle in red, the roll-out in green and the straight flight in yellow.*

where $\mu_{ri} = p_{ss} t_r$. Solved for the half roll time t_r this yields

$$t_r = t_{ri} = t_{ro} = \frac{\arccos \sqrt{\mathrm{e}^{\frac{-p_{ss} V \Delta\chi}{g}}}}{p_{ss}}, \tag{5.62}$$

which is the foundation for the following simulation method.

Figure 5.5(a) shows the limit case, when the course change is attained exactly in the planning time increment Δt, by the roll-in and roll-out segment in $2t_r$, before the maximum bank angle μ_{max} is achieved (see leftmost curve in Figure 5.6). This is the equivalent of the entirely red curve in Figure 5.2, in Section 5.2. The complete time to achieve the commanded course change is

$$t_{\Delta\chi} = \Delta t = 2t_r. \tag{5.63}$$

TS is initialized with $\mathrm{x}_{in} = (x_{in}, y_{in}, \mu_{in}, \chi_{in})$ and $p_{ri} = \pm p_{ss}$. The state at time $t_{in} + t_r$ is the final state of the roll-in $\mathrm{x}_{fin} = (x_{fin}, y_{fin}, \mu_{fin}, \chi_{fin})$, which is used to reinitialize the TS integration for the roll-out segment. The roll-out rate is set with the opposite sign as the roll-in rate $p_{ro} = -p_{ri}$. The final state of the roll-out segment in (a) is the auxiliary state x_{aux}.

In Figure 5.5(b) $t_r < t_{\mu_{max}}$ and $2t_r < \Delta t$ and therefore, an additional straight segment is added to obtain x_{aux} at $t_{in} + \Delta t$. After the roll-out segment, which is computed as before, TS is initialized with the final roll-out state and $p_{ss} = 0\ ^\circ/\mathrm{s}$. The simulation time for the straight segment is calculated by

$$t_s = \Delta t - 2t_r. \tag{5.64}$$

The final state after the straight segment in (b) is the new x_{aux}.

The cases in Figure 5.5(c) and (d) occur, when the necessary half roll time to achieve $\Delta\chi$ is greater than the time to attain the maximum bank angle $t_r > t_{\mu_{max}}$. In this case a segment with constant bank $\mu = \mu_{max}$ (red segment in (c) and (d)) has to be added in order to achieve the desired course change. Figure 5.5(c) shows the second limit case, where $\Delta\chi$ is achieved directly after the roll-out segment. The missing $\Delta\chi$, which cannot be achieved by roll-in and roll-out, is computed by

$$\Delta\chi_{nr} = \Delta\chi - 2\Delta\chi_r, \tag{5.65}$$

where $\Delta\chi_r$ is the achievable course change by roll-in and roll-out in $2t_{\mu_{max}}$. The corre-

sponding no-roll time t_{nr} is calculated by

$$\Delta\chi_{nr} = \frac{V t_{nr}}{R}, \tag{5.66}$$

with R being the turn radius

$$R = \frac{V^2}{g \tan \mu_{max}}. \tag{5.67}$$

Solved for t_{nr} this yields

$$t_{nr} = \frac{\Delta\chi_{nr} R}{V}. \tag{5.68}$$

The complete time to achieve the desired course change in Figure 5.5(c) is

$$t_{\Delta\chi} = \Delta t = 2t_{\mu_{max}} + t_{nr} \tag{5.69}$$

and the final roll-out state of the straight segment is the new x_{aux}.

If $2t_{\mu_{max}} + t_{nr} < \Delta t$, a straight segment has to be added, which corresponds to the yellow line in Figure 5.5(d). The TS simulation time for the straight segment is given by

$$t_s = \Delta t - 2t_{\mu_{max}} - t_{nr}. \tag{5.70}$$

The achievable $\Delta\chi$ is curbed by the planner's time increment Δt. If $2t_r > \Delta t$ and $t_r < t_{\mu_{max}}$ or $2t_r + t_{nr} > \Delta t$ and $t_r > t_{\mu_{max}}$, the desired $\Delta\chi$ cannot be achieved. Figure 5.6 shows results for the above described TS simulation scheme. In the example HAPS flies with $V = 22$ m/s at an altitude of 10000 m, the time increment is $\Delta t = 50$ s and the desired course change is $\Delta\chi = 45°$. The steady-state roll rate p_{ss} ranges from approximately 0.16 °/s to 3.43 °/s. The individual sections are color-coded exactly as in Figure 5.5. The lateral distance between the turns decreases with increasing roll rate. Generally, the results are accurate and the computation is fast. The mean absolute course error (desired course versus resulting course) with twelfth order TS is $4.6 \cdot 10^{-3°}$, which is remarkable, taking into account that the simulations are performed without intermediate reinitialization, completely relying on precalculated switching times. The mean runtime to compute the auxiliary states, using Matlab R2015b code on a computer with Intel Core i7-6700 CPU @ 3.40 GHz and 32 GB RAM running on 64-Bit-Windows 7, is $8.5 \cdot 10^{-5}$ s.

In case that the maximum bank angle is not attained before the desired course change (see Figure 5.5(a) and (b)), it is possible to adjust the piecewise steady-state roll rate, such that the simulation results are dynamically accurate without losing the advantage of fast runtimes. This is possible, due to an analytical determination of the previously introduced

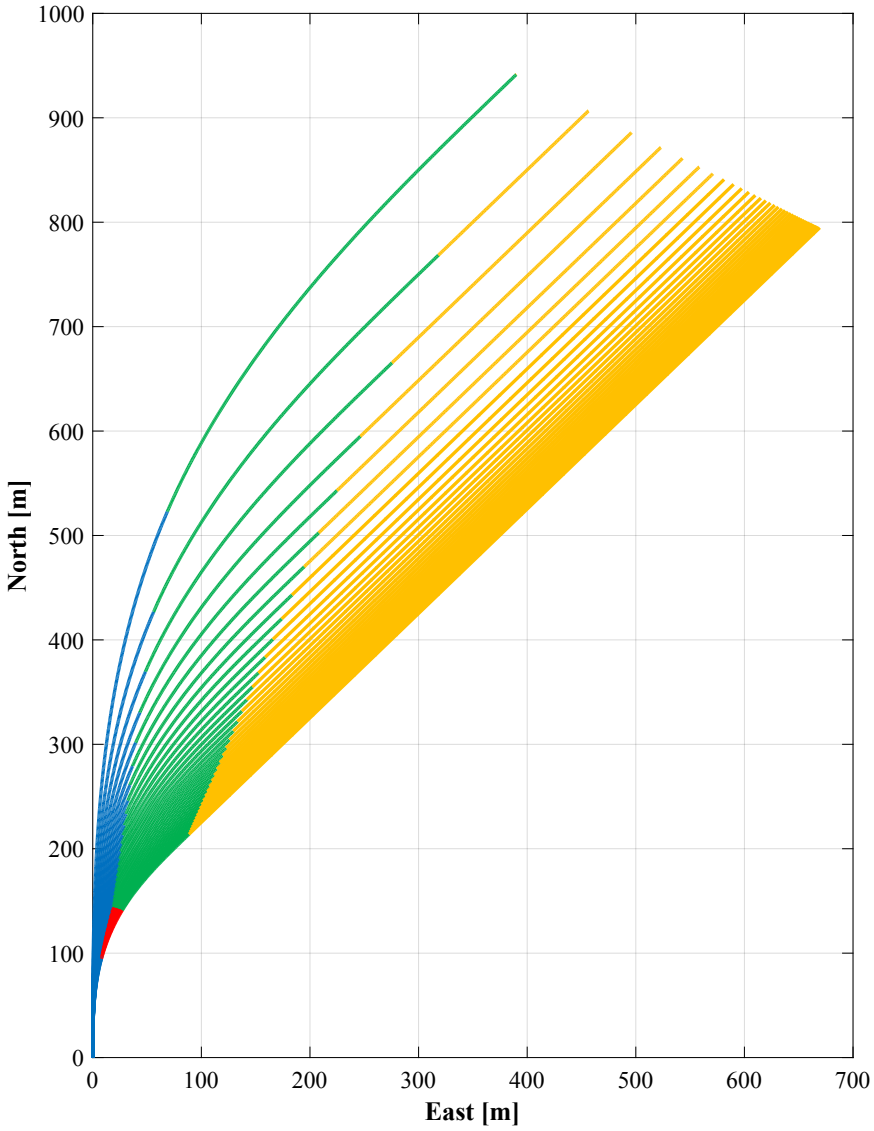


Figure 5.6: *Exemplary turns computed with TS integration, based on a simplified aircraft model, for a commanded course change of* $\Delta\chi = 45°$*. The color coding is identical to that in Figure 5.5. The initial state is* $\mathrm{x}_{in} = (0, 0, t_{in})$ *and the respective auxiliary states are located at the end of the curves at time* $t_{in} + \Delta t$*. The leftmost trajectory is the limit case, shown in Figure 5.5(a), where a turn solely consists of a roll-in (blue) and roll-out segment (green).*

half roll time t_r. It shows that it is a good approximation for the points in time, when the roll function, which considers the finite roll-acceleration by the step response function in Equation (5.56), starts to decrease or increase (gray dash-dotted lines in Figure 5.7 and 5.8), in order to achieve the commanded $\Delta\chi$. The roll rates with finite acceleration (black line in the upper plots) switch at $t \approx t_r$ and $t \approx 2t_r$. Then, it takes approximately 7τ until $p(t)$ is back to zero (gray dotted lines). Based on this knowledge, the steady-state roll-in and roll-out rate of the approximation can be calculated noniteratively, so that the resulting course change and turning-flight trajectories match reasonably. For this purpose

the integral of the exact $p(t)$ during the roll-in phase from t_{in} to $t_{in}+t_r+\tau \ln 2$ is computed by

$$\mu_{ri} = \int_{t_{in}}^{t_r} p_{ss} \left(1 - \mathrm{e}^{-t/\tau}\right) dt + \int_{t_{in}+t_r}^{\tau \ln 2} p_{ss} - 2p_{ss}\mathrm{e}^{-t/\tau}\, dt + \mu_{in}, \tag{5.71}$$

where $\mu_{in} = 0°$ due to the initial conditions and the time $\tau \ln 2$ results from solving

$$p_{ss} - 2p_{ss}(1 - \mathrm{e}^{-t/\tau}) = 0 \tag{5.72}$$

for t in order to determine the zero crossing of the exact $p(t)$. The integrals can be solved analytically by

$$\mu_{ri} = p_{ss} \left(\left(-1 + \mathrm{e}^{-t_r/\tau}\right) \tau + t_r)\right) - p_{ss}\tau \left(-1 + \ln 2\right) + \mu_{in}. \tag{5.73}$$

Therewith, the approximate steady-state roll-in rate is calculated by

$$p_{ari} = \frac{\mu_{ri}}{t_r + 2\tau \ln 2}, \tag{5.74}$$

which is always smaller than the actual steady-state rate, since due to the infinite roll acceleration assumption the achievable bank angle is generally overestimated. The same is true for the approximate steady-state roll-out rate, which is determined by

$$p_{aro} = \frac{\mu_{ri}}{t_r}. \tag{5.75}$$

The resulting approximation is acceptable for aircraft with small roll-time constants, as can be seen in Figure 5.7, where the results for HAPS with finite roll rate acceleration, roll time constant $\tau = 0.12$ s, $V_T = 22$ m/s and steady-state roll rate $p_{ss} = 1.04°$/s ($\Delta\xi = -5°$) are depicted by black lines and the approximation by red lines. The comparison of the resulting turn trajectories is depicted in Figure 5.9(a). The omitted roll acceleration can be partly compensated by adapting the steady-state roll rates, so that the final state is near to the exact trajectory. Figure 5.8 depicts the same course change for a configuration with a considerably higher roll time constant of $\tau = 1.0$ s. However, in this case a discrepancy between the exact (solid black line) and approximated trajectory (dashed colored line) is visible (Figure 5.9(b)). The simulations in Figure 5.7, 5.8 and 5.9 are computed using Matlab's `ode45`, which is an explicit 4^{th} order RK integration method. It compares the results of the 4^{th} and 5^{th} order approximation, to automatically adapt the step size h, by estimating the error to the real function, known as Dormand-Prince method [122].

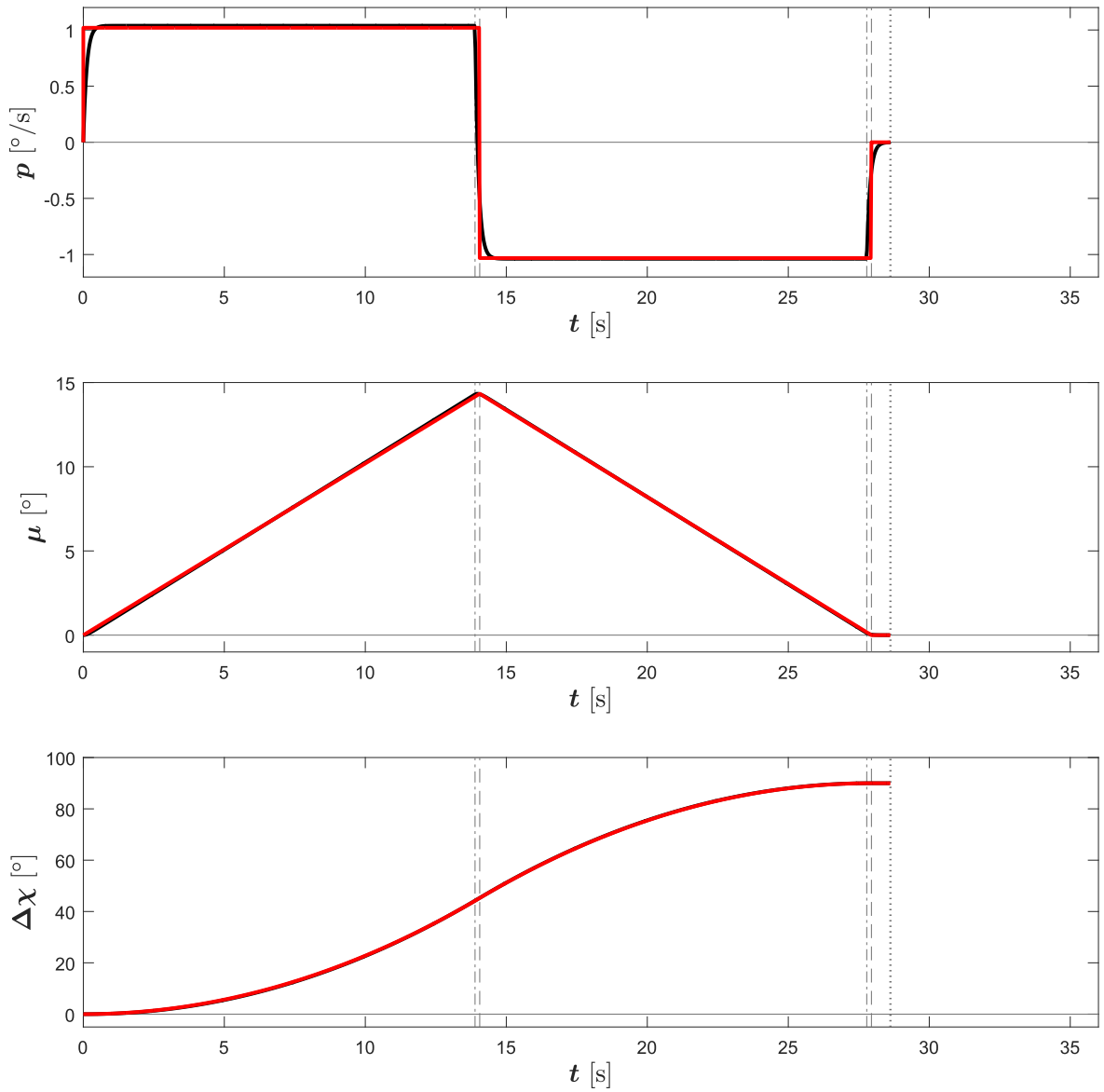


Figure 5.7: *Turn with finite roll acceleration (black lines) to achieve the commanded* $\Delta\chi = 90°$ *with HAPS versus the automated TS integration (red lines). The key to reproduce the results of the exact function is the determination of the roll time* t_r*. The agreement of the course change is very good (see bottom plot).*

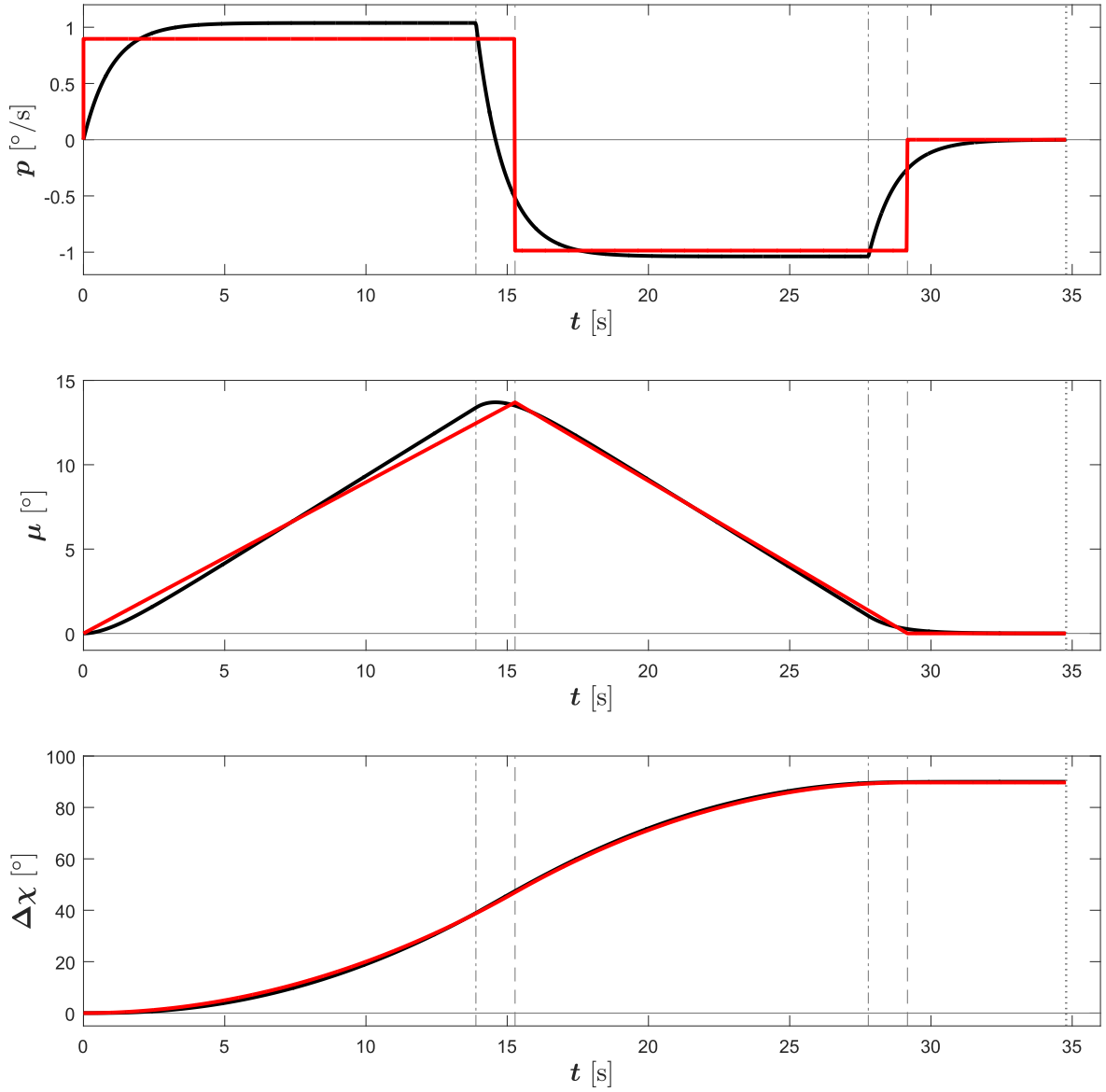
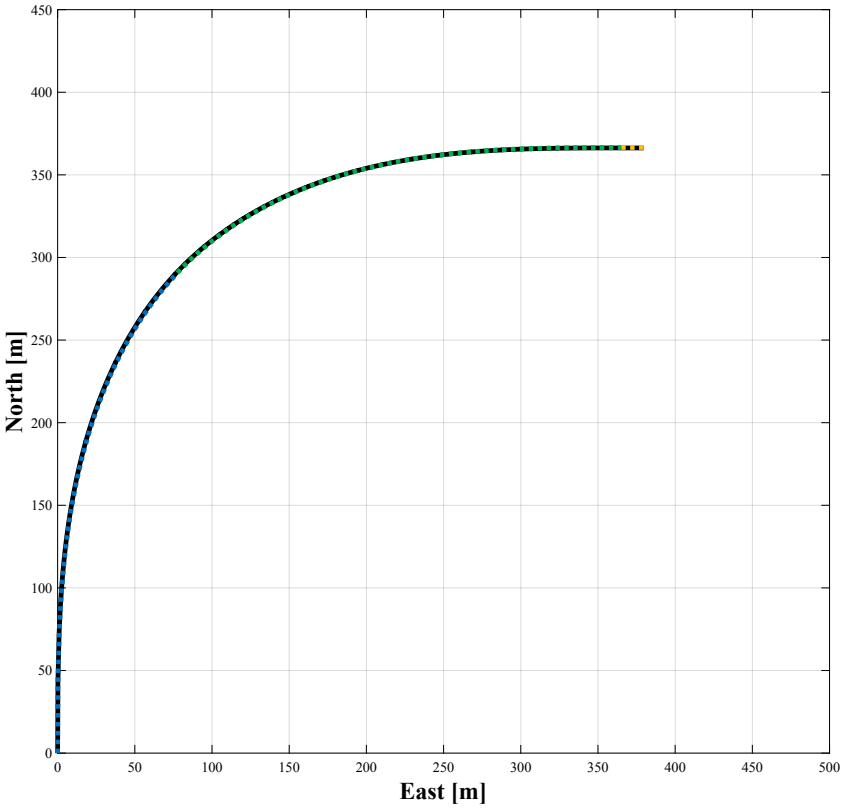
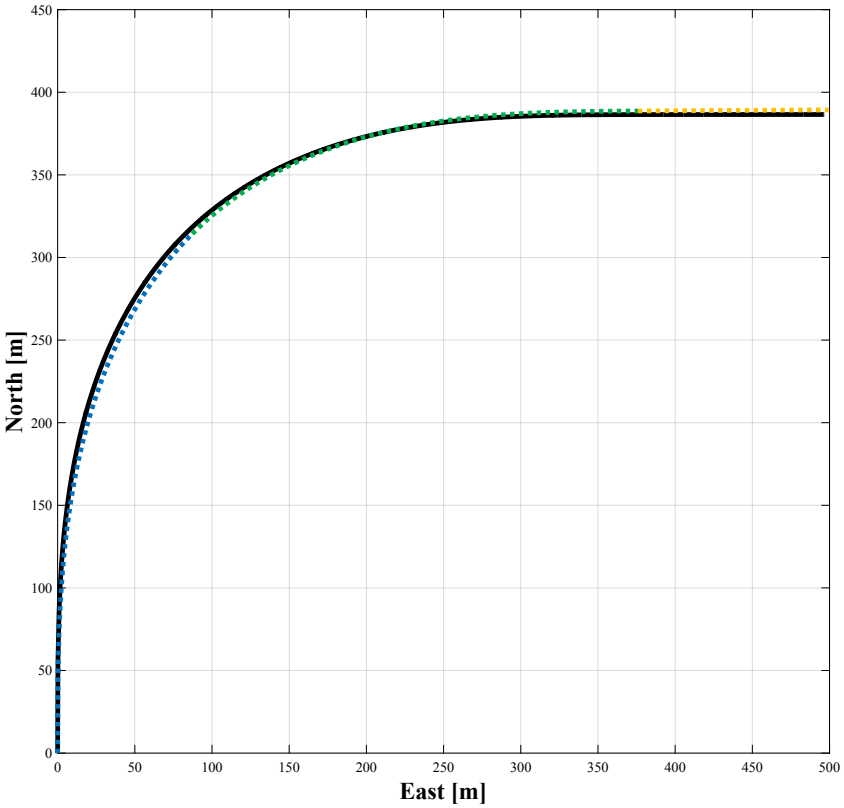


Figure 5.8: *Exact turn (black lines) to achieve a commanded* $\Delta\chi = 90°$ *versus the automated TS integration (red lines) for a roll time constant of* $\tau = 1$ *s. Roll-in and roll-out rate of the approximation are considerably different in order to match the results of the exact function.*



(**a**) *Trajectory comparison for HAPS with* $\tau = 0.12$ *s.*



(**b**) *Trajectory comparison for* $\tau = 1.0$ *s.*

Figure 5.9: *Comparison between trajectories that are computed with finite (black lines) and infinite roll rate acceleration with TS (colored lines), for (a)* $\tau = 0.12$ *s and (b)* $\tau = 1.0$ *s,* $V = 22$ *m/s and a commanded course change of* $\Delta\chi = 90°$*. The trajectories for the small roll time constant are almost congruent.*

5.4 Automatic Planning of Holding Patterns

This section features Contribution 7: *Implicit Planning of Holding Patterns.*

According to the NATO Standardization Agreement No.4586 the circle is an authorized holding maneuver for unmanned aircraft. Radius, turning-sense and duration of the hold are unrestricted [123]. Before a trajectory is planned the algorithm determines if the goal will be covered by obstacles in the period from t_0 to $t_0 + T_N$. If the goal is permanently covered during the period of the actual nowcast, it is declared as not flyable and the aircraft performs a hold in X_{free} until the next nowcast is issued. It is possible to specify holding locations in advance, which the planner can use when they are in X_{free}. If the goal is only temporarily covered, a holding pattern is performed until the goal is in EX_{free}. Therefore, an unidirectional turning constraint is applied, for example only turn right (UPS-method). This prevents the MPTP to plan meandering trajectories, however, they may not be near-optimal anymore. It is also important to notice, that resulting trajectories can be considerably different, depending on the selected sense of rotation. The x_{aux}, which are always added to model the nonholonomic turning-flight constraint, naturally result in circular holding patterns in EX_{free}, if the goal is covered. The ability to plan holding patterns therefore exists innately, which is very practical and keeps the algorithm simple. By setting $|\Delta\chi_{max}| = 45°$, using the method described in Section 5.1, a circular holding pattern is described by the x_{aux}, as inscribed regular n-gon with $n = 8$ (see Figure 5.10, Section 6.3). The distance on a straight segment of an n-gon is given by $S_n = V_T \Delta t$. Therefore, a full n-gon takes $n\Delta t$ time. For fly-over states, the radius R_{cc} of the corresponding circumcircle, on whose orbit the aircraft flies, is

$$R_{cc}(n) = \frac{S_n}{\sqrt{2 - 2\cos(2\pi/n)}}. \tag{5.76}$$

The distance on the corresponding circumcircle segment is

$$S_{cc}(n) = R_{cc}\Delta\chi_{max} = R_{cc}2\pi/n. \tag{5.77}$$

As $S_n < S_{cc}$, the velocity has to be corrected, in order for the aircraft to arrive at a state in due time. The correction factor is calculated by the relative error of the distances to S_n, as Δt is a fixed value

$$\delta V(n) = 1 + \frac{S_{cc} - S_n}{S_n} = 1 + \frac{2\pi/n - \sqrt{2 - 2\cos(2\pi/n)}}{2\pi/n}. \tag{5.78}$$

The mean corrected velocity is calculated by

$$V_{cor} = V_T \delta V(n = 8) = 80 \cdot 1.026 \text{ m/s} = 82.04 \text{ m/s}. \tag{5.79}$$

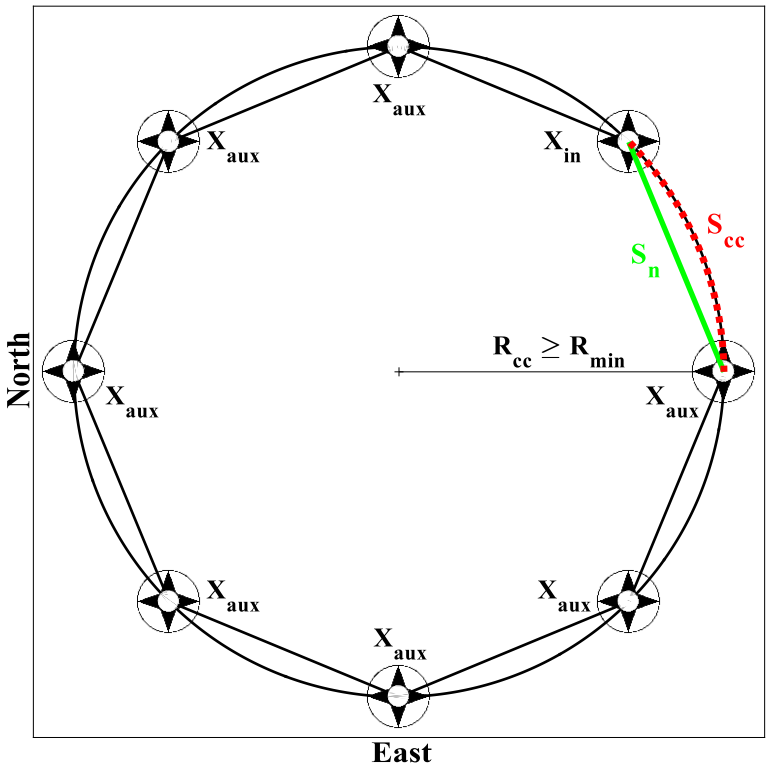


Figure 5.10: *Circular holding pattern, defined by eight fly-over states.*

A standard-rate-turn or rate-one-turn (ROT) takes $t_{ROT} = 120$ s to complete a full 360° turn. In order to determine the appropriate time increment to perform a ROT, the following steps are necessary. The circumference of the circumcircle for a ROT is $V_{cor} t_{ROT}$, which for a planning velocity of 82.04 m/s yields $C_{cc} = 9844.84$ m. The corresponding radius is

$$R_{cc} = \frac{C_{cc}}{2\pi} = 1566.86 \ m. \tag{5.80}$$

By rearranging the Equation (5.76) the straight segment distance is

$$S_n = R_{cc}\sqrt{2 - 2\cos(2\pi/n)}. \tag{5.81}$$

Using the relationship $S_n = V_T \Delta t$, the equation can be written as

$$V_T \Delta t = R_{cc}\sqrt{2 - 2\cos(2\pi/n)}. \tag{5.82}$$

Solved for the time increment this yields

$$\Delta t = \frac{R_{cc}\sqrt{2 - 2\cos(2\pi/n)}}{V_T} = \frac{1566.86 \text{ m}\sqrt{2 - 2\cos(2\pi/8)}}{80 \text{ m/s}} = 14.99 \text{ s}. \tag{5.83}$$

For the interpolation of nowcast data (see Section 3.1.6) it is necessary that Δt is a factor of the nowcast update interval $\Delta T_N = 300s$. In this case, the nearest factor is $\Delta t = 15$ s. Finally, it is important to evaluate, if the bank angle for the ROT is admissible. The corrected velocity is $V_{cor} = 82.04$ m/s and the radius of the circumcircle is 1566.86 m. Therefore, the required bank angle in this example is

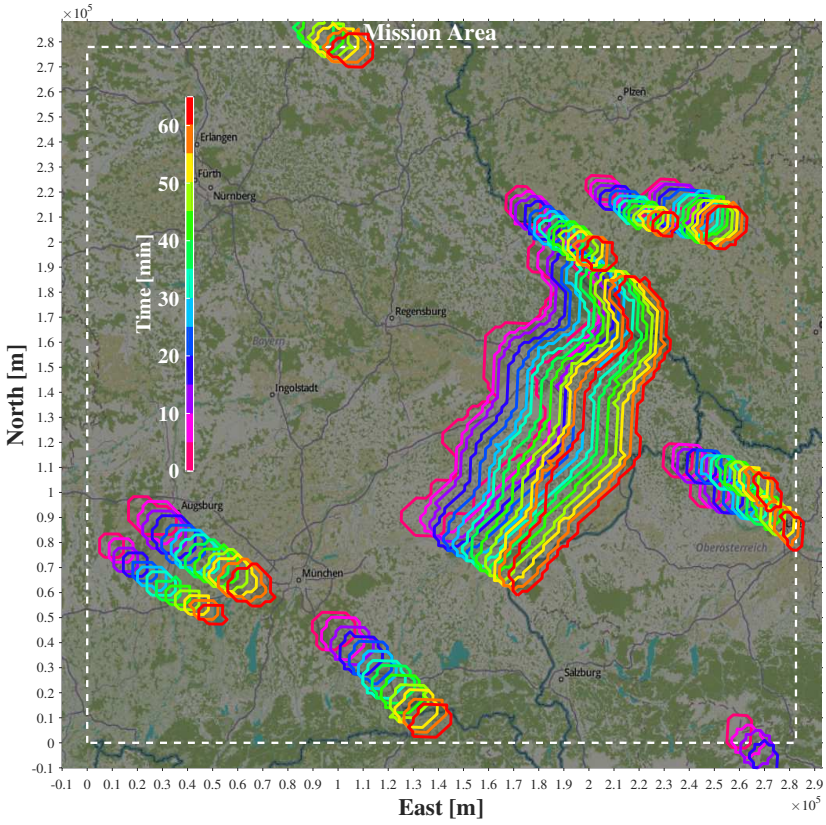
$$\mu_{ROT} = \arctan \frac{V_{cor}^2}{gR_{cc}} = \arctan \frac{(82.04 \ m/s)^2}{9.81 \text{ m/s}^2 \ 1527.89\text{m}} = 23.67°. \tag{5.84}$$

If $\mu_{ROT} \leq \mu_{max}$ and $V_S \leq V_{cor} \leq V_{T_{max}}$, the flight controller is able to put the presented guidance strategy into practice in the wind-free case, arriving at the commanded states in due time. If the method for x_{aux} generation from Section 5.3 is applied, the transitions are smooth and the feasibility is ensured. Generally, the use of Euclidean distance heuristic (EDH) is best suited for planning of holding patterns, as the SSHP requires the initial state to be connected to the goal state, at least by a partial shortest path map (PSPM) (see Section 4.1.2). Since the goal state is temporarily covered, this is not possible without edges penetrating EX_{obs}. A possible workaround is to multiply the weights of intersecting edges, by a sufficiently large penalty value, so that an actually longer trajectory is the shortest. However, it is difficult to determine the penalty in such way, that the optimal trajectory is computed. Most of the trajectories, calculated in this way, are neither intuitive nor optimal.

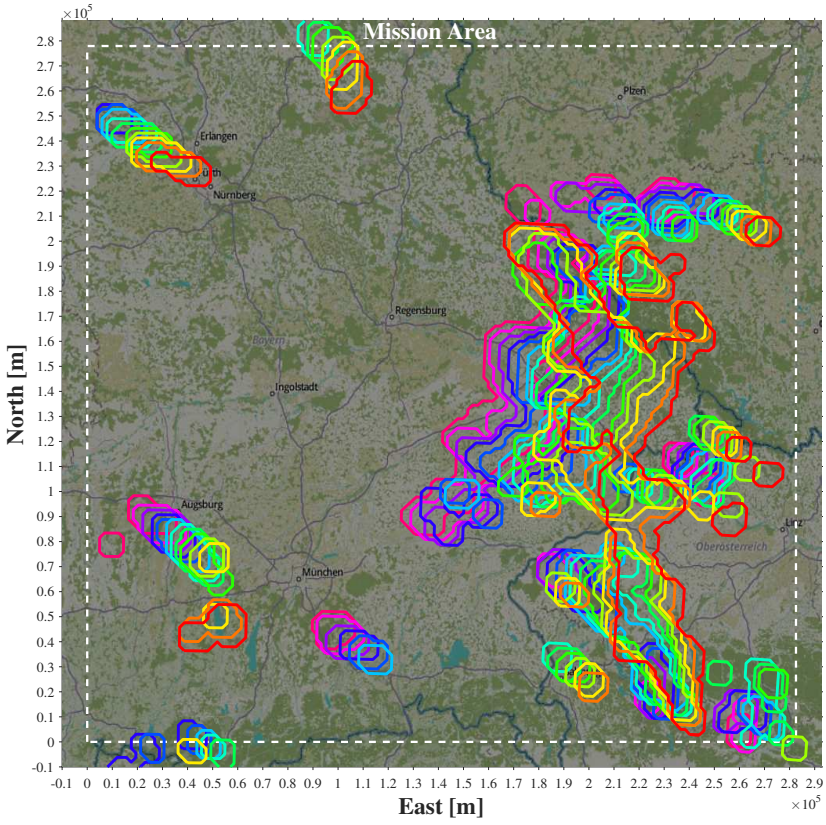
Chapter 6

Results

This chapter presents simulation results for the main capabilities of the model predictive trajectory planner (MPTP). The results of the MPTP subsystems are found in the respective sections. All simulations are computed using Matlab R2015b code on a computer with Intel Core i7-6700 CPU @ 3.40 GHz and 32 GB RAM running on 64-Bit-Windows 7. As the planning algorithm always performs the same calculation steps and number of iterations, the computational times are averaged over ten runs. The mission area (white dashed lines in Figures 6.1 - 6.17) extends from 47.50° to 50.00° in latitude and 10.25° to 14.00° in longitude. The origin of the coordinate system in Figures 6.1 to 6.17 is at 47.50° latitude and 10.25° longitude. It is assumed, that the wind can be compensated by the flight controller, as an external source of disturbance may prevent the results by different heuristic methods from being comparable. However, the upcoming Figure 6.4 depicts a series of trajectories, which are computed considering a constant wind, and thereby shows that the MPTP is able to handle it. The maximum bank angle of the aircraft is $|\mu_{max}| = 25°$ and the maximum true airspeed is $V_{T_{max}} = 90$ m/s. The performance data matches with the aircraft presented in Section 5.3.1. For the following two avoidance scenarios, thirteen historical nowcasts by Rad-TRAM from 06/27/2015 are used. All figures have the same color-code for real time. The period from t_0 =19:05 h to 20:05 h UTC is selected, due to the significant divergence, between nowcast and measured weather. Figure 6.1(a), shows the nowcast issued at t_0, in the nowcast period of t_0 to $t_0 + 60$ minutes, while Figure 6.1(b) shows the respectively measured data for the same instants.



(**a**) *Nowcast for one hour.*



(**b**) *Actual measurement for the hour.*

Figure 6.1: *(a) Thunderstorm prediction by the nowcast, issued at 19:05 h, for the next hour and (b) subsequent real measurements for the same period of time, both issued by Rad-TRAM.*

6.1 Continuous Avoidance using Different Heuristics

In this section the results of the A*-search using the novel SSPH heuristic (introduced in Section 4.3.3) are compared to those of A*-search using Euclidean distance heuristic (Section 4.3.2) and Dijkstra's algorithm (Section 4.3.1). The latter two are guaranteed to find the optimal solution, if one exists.

An anticipatory trajectory is computed based on the expectation of how the future situation will look like (Section 3.3). The Figure 6.2 shows an exemplary trajectory based on the nowcast from t_0 =19:05 h (see Figure 6.1(a)). It is the first trajectory of the continuous avoidance example in Figure 6.5 (magenta line). The vertical dimension is the

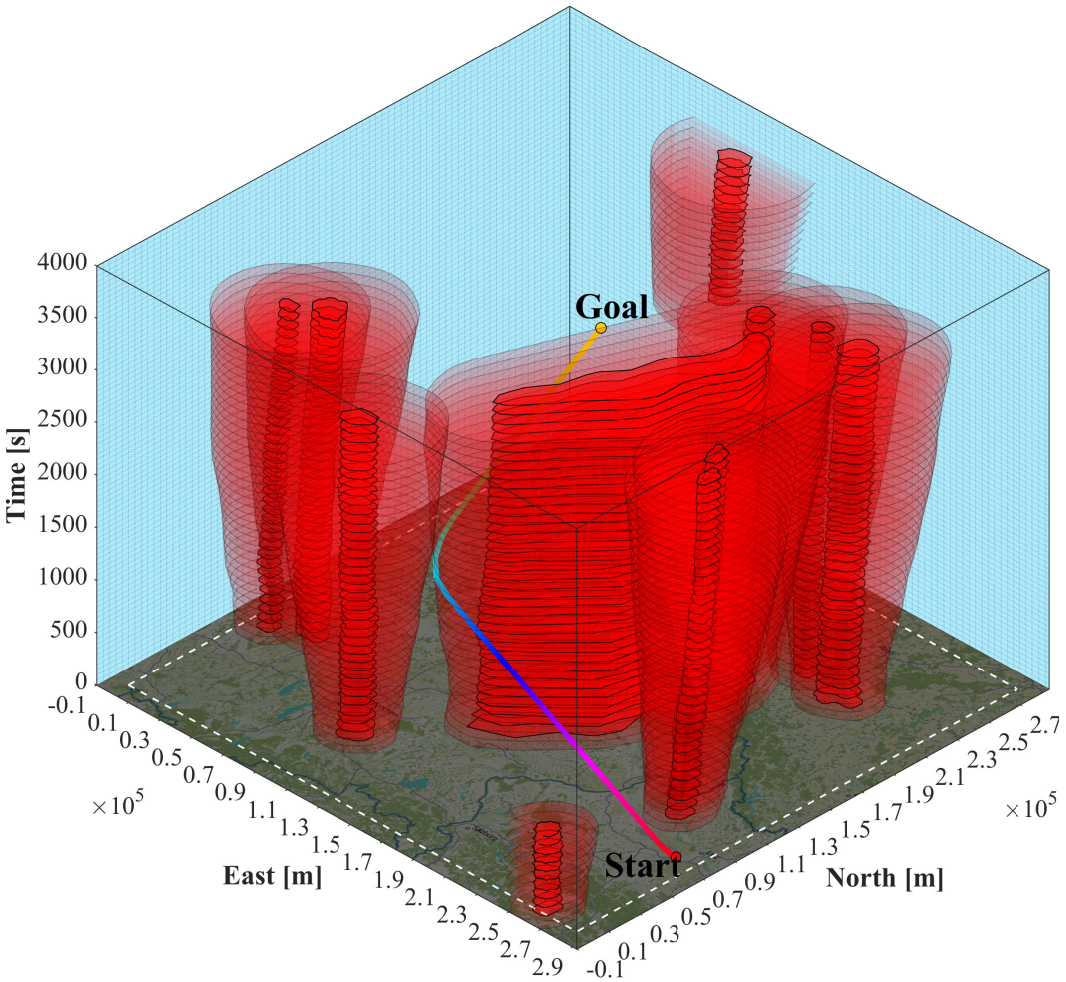


Figure 6.2: *Near-optimal anticipatory avoidance trajectory, based on a nowcast issued at 19:05 h.*

nowcast time T_N. The dark red areas show the prediction of the nowcast for the next hour. The surrounding light red areas are the sum of safety and probabilistic margins, introduced in Section 3.1. Their size varies between 10000 m at t_0 to 35000 m at $t = T_N = 3600$ s and defines the regions to be avoided. The aircraft flies with constant $V_T = V_G = 80$ m/s.

Future states are sampled on a circular grid with $\Delta\chi = 2°$ using the approximate method from Section 3.2. Continuous avoidance is achieved by recurrent replanning, based on the latest environment prediction.

Therefore, more examples of anticipatory trajectories can be found in Figures 6.3 and 6.5 (colored lines), which show the results for the following two continuous avoidance scenarios, in which the aircraft has to fly towards developing thunderstorms in order to reach the goal. Both scenarios are critical cases as uncertainty in the nowcast is generally most pronounced in moving direction of a storm [70]. None of the trajectories is post-processed, e.g. by B-splines. The colored lines in Figures 6.3 and 6.5 are anticipatory trajectories, that are computed by the MPTP, starting from the triangles of corresponding color. They mark intermediate states, from which the MPTP is reinitialized. The continuous trajectory (polychrome line along the triangles) is the composite of the first leg traveled in ΔT_N on the successive anticipatory trajectories. It is important to notice that only the colors of the continuous trajectory match with those of the real thunderstorm measurements.

Since all conditions for feasibility from Sections 5.1 and 5.4 are fulfilled, the assumption is made that commanded states are reached in due time, so that a next initial state (see Section 3.2) is part of the previously computed trajectory. Hence flight controller and aircraft blocks are not modeled. While this setup is not eligible to prove the feasibility of the proposed guidance strategy, it is appropriate to explain the presented trajectory planning algorithm (see Section 3 and 4), demonstrate continuous avoidance trajectories on the basis of real thunderstorm forecasts, and compare different computation methods. The feasibility of the proposed guidance strategy, using a controlled 6-DoF aircraft model, has been demonstrated in [3].

The trajectories in Figures 6.3 and 6.5 are computed with A*-search using EDH and SSPH (Section 4.3) and serve for optical comparison. The runtime for each MPTP iteration is plotted in the legends on the top left side. The runtimes for the weather processing depend on the data and take around $0.9 - 2.5$ s without clustering. They are not included in the runtimes for trajectory planning as the weather processing is done in advance (see Chapter 2). Additionally, a constant H-cost of zero is applied, which degrades the A*-search to the Dijkstra-algorithm. Random perturbations of aircraft states are intentionally omitted in order to compare the different methods, regarding their convergence and resulting trajectories. A trajectory is path parameterized with time and therefore consists of states which are abbreviated as $TRST$. The states which are explored by A*-search are called $EXST$. The relation between the number of $TRST$ and $EXST$ is

a measure for the efficiency of the search. Two avoidance scenarios are presented in the following. Each scenario is computed with unidirectional or bidirectional turning-flight (Chapter 5). Tables 6.2 to 6.16 list the results for trajectory planning, with the different heuristics, for every iteration of the MPTP. For the nonholonomic turning-flight the method from Section 5.1 is applied.

6.1.1 Scenario 1

Table 6.1 lists the parameters for the simulation of the first scenario.

Table 6.1: *Parameters for the first continuous weather avoidance scenario. Index s stands for start state and index g for the goal state.*

$\boldsymbol{\varphi_s}$	48.83°
$\boldsymbol{\lambda_s}$	13.54°
$\boldsymbol{\chi_s}$	280°
$\boldsymbol{\varphi_g}$	49.01°
$\boldsymbol{\lambda_g}$	11.98°
$\boldsymbol{t_0}$	19 : 05 h
$\boldsymbol{\Delta t}$	100 s
$\boldsymbol{V_T}$	80 m/s
$\boldsymbol{\lvert\Delta\chi\rvert}$	45°
$\boldsymbol{\epsilon}$	0.022°

The resulting trajectory lengths in scenario 1 are identical for all MPTP iterations, when using Dijkstra and A*-search with EDH. However, only nine out of ten trajectories have identical lengths, when using A*-search with SSPH, which is why the lengths are listed in the tables below. In the fourth MPTP iteration the trajectory using SSPH is about 200 m longer than with the other methods. The reason for this discrepancy and the following results are discussed in Section 7.

Unidirectional Turning-Flight

The results in Figures 6.3 to 6.5 and Tables 6.2 to 6.4 are computed with turning constrained to right-sense. This means, that in the main loop of A*-search (see Section 4.2), for every explored state an x_{aux} to the right is added to the OL, as described in Chapter 5. The MPTP is able to consider constant wind (speed and direction independent of time and space), using the approximate future aircraft states, introduced in Section 3.2 (see Figure 3.8). In the example shown in Figure 6.4 the mean wind comes from north at a speed of 10 m/s, which is why initially the aircraft's ground speed is lower as it encounters headwind. In return it subsequently has to deviate less (compared to Figure 6.3)

and arrives, due to a slight tailwind, at approximately the same time at the goal as in Figure 6.3.

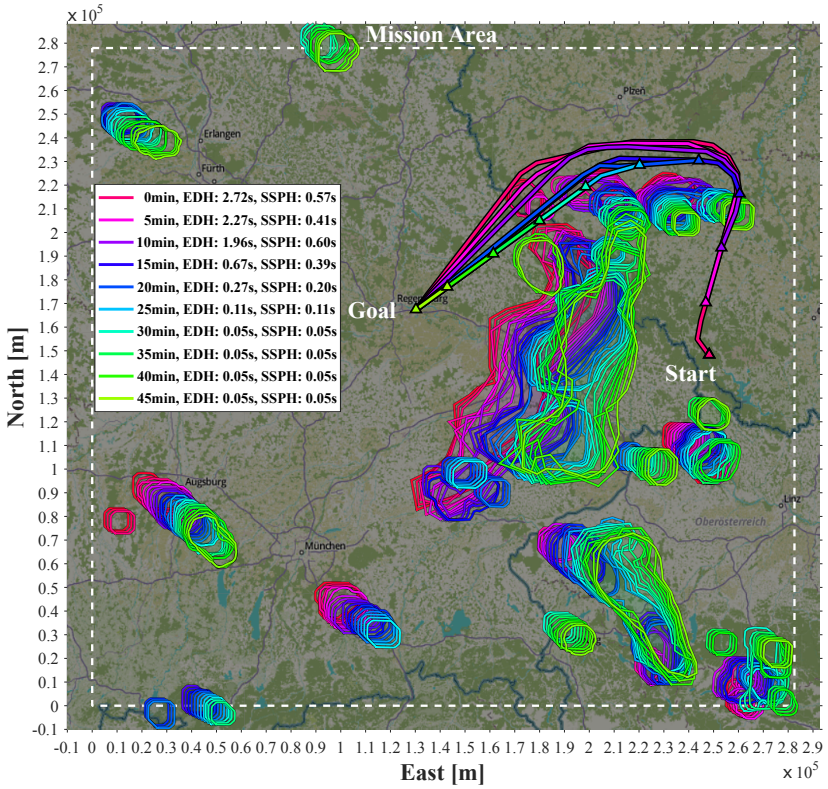


Figure 6.3: *All the anticipatory trajectories for the first scenario, computed by A* with EDH and SSPH, are plotted. Although the fourth trajectory (dark blue) is slightly longer when using SSPH, the difference is not visible. The runtimes for all MPTP iterations with EDH and SSPH are listed in the legend.*

Table 6.2: *Dijkstra's algorithm applied to the avoidance in Figure 6.3, if the aircraft can only turn right.*

MPTP Iter.	TL [km]	TRST [-]	EXST [-]	TRST/EXST [%]
1	247.0	13	628	2.070
2	219.6	11	710	1.549
3	192.6	9	996	0.904
4	159.5	9	420	2.143
5	134.5	2	230	0.870
6	108.6	2	96	2.083
7	84.80	1	115	0.870
8	61.70	1	8	12.50
9	38.60	1	5	20.00
10	15.40	1	2	50.00

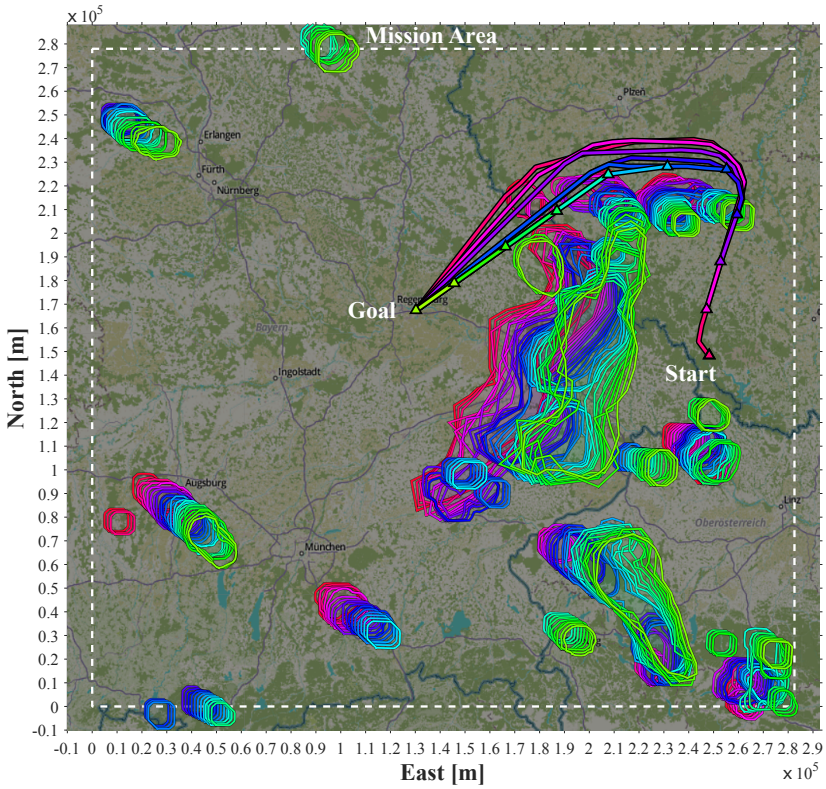


Figure 6.4: *All anticipatory trajectories for the first scenario considering a constant mean wind. Due to the distorted isochronous aircraft progress the planned trajectories are optimal with respect to the actual wind conditions.*

Table 6.3: *A* with EDH applied to the avoidance in Figure 6.3, for right turn only.*

MPTP Iter.	TL [km]	TRST [-]	EXST [-]	TRST/EXST [%]
1	247.0	13	58	22.41
2	219.6	11	49	22.45
3	192.6	9	50	18.00
4	159.5	9	14	64.29
5	134.5	2	5	40.00
6	108.6	2	2	100.0
7	84.80	1	1	100.0
8	61.70	1	1	100.0
9	38.60	1	1	100.0
10	15.40	1	1	100.0

Bidirectional Turning-Flight

Here, the aircraft can turn left and right, which ensures, that the trajectory contains no unnecessary loops. Tables 6.6 to 6.7 contain results for the case, that in the while loop of the A*-search (see Section 4.2) for every explored state two x_{aux} to the right and left are added to the OL, as described in Chapter 5. Dijkstra is not listed for comparison as even the first MPTP iteration does not converge after several hours and 1.5×10^6 iterations.

Table 6.4: *A*-search with SSPH applied to the avoidance in Figure 6.3, if the aircraft can only turn right.*

MPTP Iter.	**TL** [km]	**TRST** [-]	**EXST** [-]	**TRST/EXST** [%]
1	247.0	13	13	100.0
2	219.6	11	11	100.0
3	192.6	9	14	64.29
4	159.7	6	6	100.0
5	134.5	2	3	100.0
6	108.6	2	2	100.0
7	84.80	1	1	100.0
8	61.70	1	1	100.0
9	38.60	1	1	100.0
10	15.40	1	1	100.0

Table 6.5: *Ratios between explored states for Dijkstra vs. A*-search with SSPH and A*-search with EDH vs. A*-search with SSPH, if the aircraft can only turn right.*

MPTP Iter.	**EXST**(Dijkstra)/**EXST**(SSPH) [-]	**EXST**(EDH)/**EXST**(SSPH) [-]
1	48.31	4.462
3	71.14	3.571
4	70.00	2.333
5	76.67	1.667
6	48.00	1.000
7	115.0	1.000
8	8.000	1.000
9	5.000	1.000
10	2.000	1.000

Table 6.6: *A*-search using EDH applied to the avoidance in Figure 6.3, if the aircraft can turn left and right.*

MPTP Iter.	**TL** [km]	**TRST** [-]	**EXST** [-]	**TRST/EXST** [%]
1	247.0	13	12159	0.107
2	219.6	11	1397	0.787
3	192.6	9	113	7.964
4	159.5	9	14	64.29
5	134.5	2	5	40.00
6	108.6	2	2	100.0
7	84.80	1	1	100.0
8	61.70	1	1	100.0
9	38.60	1	1	100.0
10	15.40	1	1	100.0

Table 6.7: *A*-search using SSPH applied to the avoidance in Figure 6.3, if the aircraft can turn left and right.*

MPTP Iter.	**TL** [km]	**TRST** [-]	**EXST** [-]	**TRST/EXST** [%]
1	247.0	13	15	86.67
2	219.6	11	11	100.0
3	192.6	9	14	64.29
4	159.7	6	6	100.0
5	134.5	2	3	66.67
6	108.6	2	2	100.0
7	84.80	1	1	100.0
8	61.70	1	1	100.0
9	38.60	1	1	100.0
10	15.40	1	1	100.0

Table 6.8: *Ratios between explored states for A*-search with EDH vs. A*-search with SSPH, if the aircraft can turn left and right.*

MPTP Iter.	**EXST**(EDH)/**EXST**(SSPH) [-]
1	810.6
2	127.0
3	8.071
4	2.333
5	1.667
6	1.000
7	1.000
8	1.000
9	1.000
10	1.000

6.1.2 Scenario 2

Table 6.9 lists the parameters for the simulation of scenario 2. In this example, identical parameters produce identical anticipatory trajectories for all MPTP iterations, when applying Dijkstra, A*-search with EDH and A*-search with SSPH. Their respective lengths are 250.5 km, 228.8 km, 199.6 km, 177.5 km, 143.7 km, 119.5 km, 92.00 km, 66.90 km, 41.80 km and 16.70 km. The results are discussed in Section 7.

Unidirectional Turning-Flight

The results in Tables 6.10 to 6.12 are computed with turning constrained to right-sense. This means, that in the main loop of the A*-search (see Section 4.2), for every explored state an x_{aux} to the right is added to the OL, as described in Chapter 5.

A comparison for the first MPTP iteration in Tables 6.11 and 6.12 is visualized in

Table 6.9: *Parameters for the second continuous weather avoidance scenario. Index s stands for start state and index g for the goal state.*

φ_s	48.154°
λ_s	13.868°
χ_s	205°
φ_g	49.006°
λ_g	11.980°
t_0	19 : 05 h
Δt	100 s
V_T	80 m/s
$\lvert\Delta\chi\rvert$	84°
ϵ	0.031°

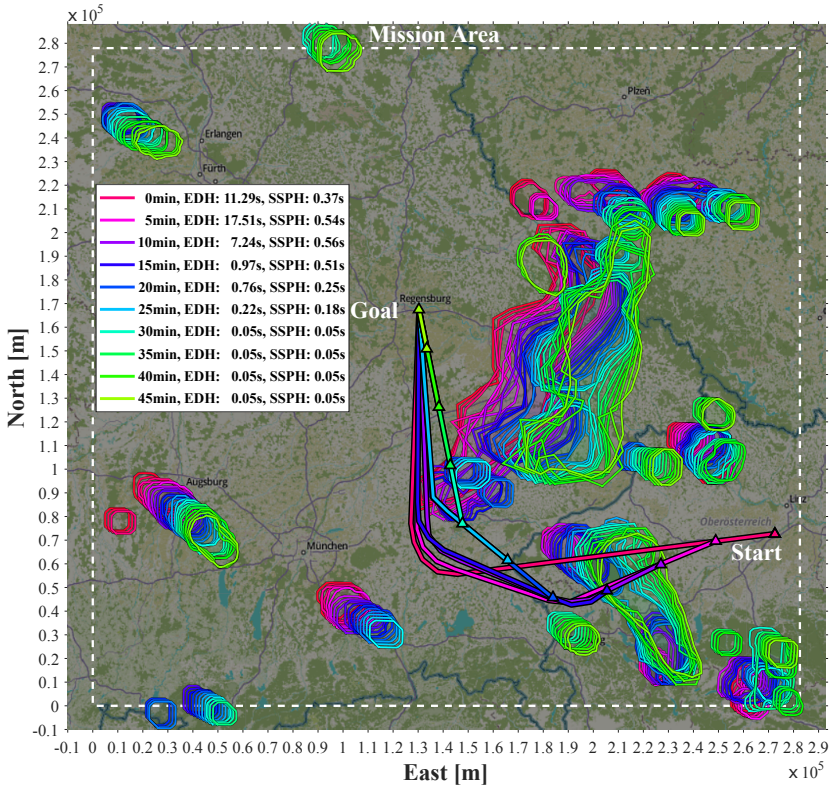
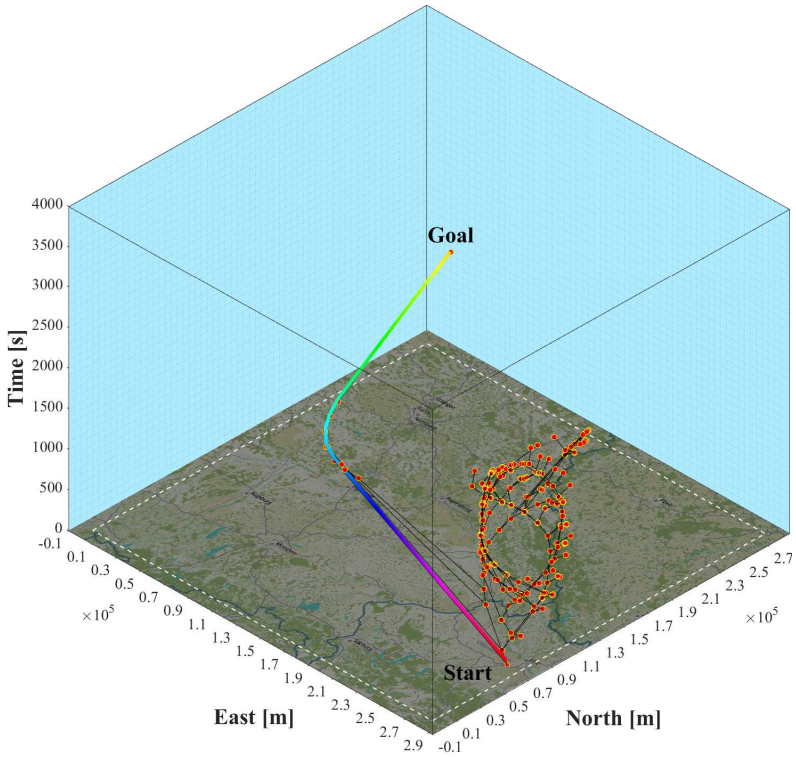
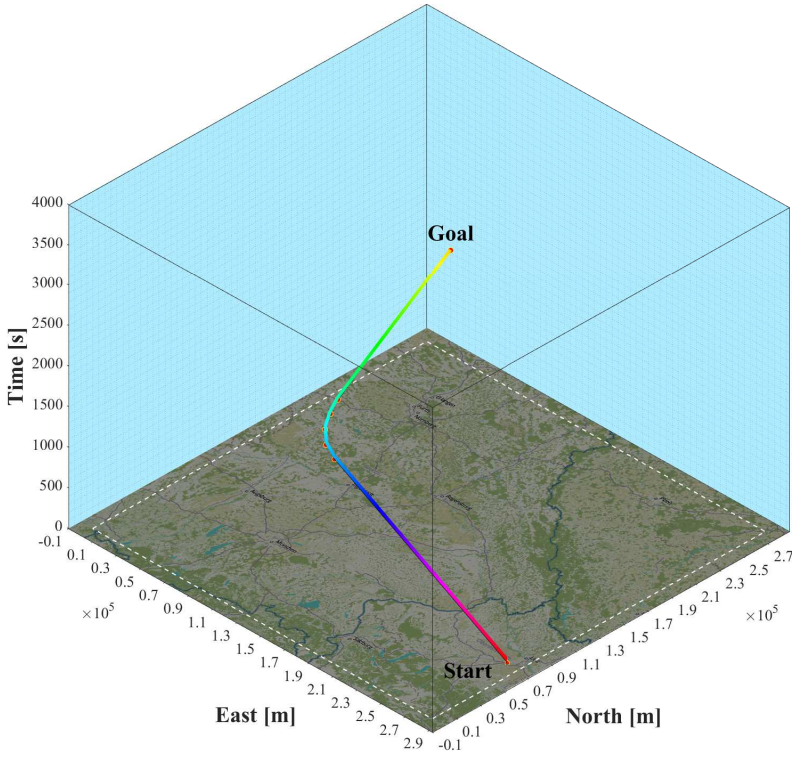


Figure 6.5: *All trajectories in the second scenario are identical for EDH and SSPH. The runtimes for the MPTP iterations are listed in the legend. The first three iterations display the strength of the SSPH. A comparison of the runtimes for the first three iterations clearly shows the advantage of SSPH over EDH.*

Figure 6.6. The success rate (TRST/EXST ratio) or rather convergence of A*-search using EDH in (a) versus the proposed SSPH in (b) is compared. The fact, that all candidate states are in the final trajectory show how accurate the cost estimation by SSPH can be, even in dense scenarios.



(**a**) *A*-search using Euclidean distance heuristic.*



(**b**) *A*-search using shortest static path heuristic.*

Figure 6.6: *(a) Trajectory with A*-search using EDH. Candidate future states in the closed list are depicted by yellow and red dots. Until the goal is reached the search incrementally generates ramifications in the X-space, all rooted in the start state. (b) Trajectory with A*-search using SSPH. The hit rate in this first MPTP iteration is* 100 *%, as all candidate states are part of the planned trajectory. This is due to an accurate estimation of the costs-to-go, which is achieved by explicitly taking obstacles into account, and represents the great strength of the new heuristic method.*

Table 6.10: *Dijkstra applied to the avoidance in Figure 6.5, if the aircraft can only turn right.*

MPTP Iter.	TRST [-]	EXST [-]	TRST/EXST [%]
1	6	17633	0.034
2	8	63940	0.013
3	8	174137	0.005
4	8	6960	0.115
5	3	768	2.391
6	3	145	2.069
7	1	24	4.167
8	1	19	5.263
9	1	6	16.67
10	1	3	33.33

Table 6.11: *A*-search with EDH applied to the avoidance in Figure 6.5, if the aircraft can only turn right.*

MPTP Iter.	TRST [-]	EXST [-]	TRST/EXST [%]
1	6	231	2.597
2	8	346	2.312
3	8	145	5.517
4	8	18	44.44
5	3	13	23.08
6	3	4	75.00
7	1	1	100.0
8	1	1	100.0
9	1	1	100.0
10	1	1	100.0

Table 6.12: *A*-search with SSPH applied to the avoidance in Figure 6.5, if the aircraft can only turn right.*

MPTP Iter.	TRST [-]	EXST [-]	TRST/EXST [%]
1	6	6	100.0
2	8	10	80.00
3	8	11	72.73
4	8	9	88.89
5	3	4	75.00
6	3	3	100.0
7	1	1	100.0
8	1	1	100.0
9	1	1	100.0
10	1	1	100.0

Bidirectional Turning-Flight

The aircraft can turn left and right, which ensures, that a trajectory does not contain unnecessary loops. The Tables 6.14 to 6.16 show results for the case, that in the main loop

Table 6.13: *Ratios between explored states for Dijkstra vs. A*-search with SSPH and A*-search with EDH vs. A*-search with SSPH, if the aircraft can only turn right.*

MPTP Iter.	**EXST**(Dijkstra)/**EXST**(SSPH) [-]	**EXST**(EDH)/**EXST**(SSPH) [-]
1	2939	38.50
2	6394	34.60
3	15830	13.18
4	733.3	2.000
5	192.0	3.250
6	48.33	1.333
7	24.00	1.000
8	19.00	1.000
9	6.000	1.000
10	3.000	1.000

of the A*-search, for every explored state, two x_{aux} are added to the OL (see Chapter 5). Dijkstra's algorithm is again not listed for a comparison for bidirectional flight, as even the first MPTP iteration does not converge after hours.

Table 6.14: *A*-search with EDH applied to the avoidance in Figure 6.5, if the aircraft can turn left and right.*

MPTP Iter.	**TRST** [-]	**EXST** [-]	**TRST/EXST** [%]
1	6	28007	0.021
2	8	13809	0.058
3	8	1267	0.631
4	8	67	11.94
5	3	19	15.79
6	3	5	60.00
7	1	1	100.0
8	1	1	100.0
9	1	1	100.0
10	1	1	100.0

Table 6.15: *A*-search with SSPH applied to the avoidance in Figure 6.5, if the aircraft can turn left and right.*

MPTP Iter.	**TRST** [-]	**EXST** [-]	**TRST/EXST** [%]
1	6	6	100.0
2	8	10	80.00
3	8	11	72.73
4	8	9	88.89
5	3	4	75.00
6	3	3	100.0
7	1	1	100.0
8	1	1	100.0
9	1	1	100.0
10	1	1	100.0

Table 6.16: *Ratios between explored states of A*-search with EDH vs. A*-search with SSPH, if the aircraft can turn left and right. Yielding the same results, the additional computational expenditure of using EDH is considerable.*

MPTP Iter.	**EXST**(EDH)/**EXST**(SSPH) [-]
1	4668
2	1381
3	115.2
4	7.444
5	4.750
6	1.667
7	1.000
8	1.000
9	1.000
10	1.000

6.2 Analysis of MPTP Performance Characteristics

In this section performance characteristics of the MPTP, i.e. mean success rate and mean search efficiency, are estimated based on a large number of samples, which are generated by performing a Monte Carlo simulation (MCS). The MCS is based on the law of large numbers and the central limit theorem. If a sample mean exists and the variance is bounded, the law of large numbers states that the mean value of independent samples converges to the true value as the number of trials goes to infinity

$$\lim_{n\to\infty} \bar{x} = \mu, \tag{6.1}$$

where n is the number of samples, $\bar{x}$ is the sample mean and μ is the true unknown mean. The central limit theorem states that given a sufficiently large number of random samples,

the distribution of the sample means is approximately normal [124].

A total of $n = 10000$ independent trajectory samples are computed by the MPTP, using EDH and the novel SSPH, for bidirectional turning-flight. The parameters for MCS are listed in Table 6.17 and the applied thunderstorm data is the Rad-TRAM nowcast from 06/27/2015, issued at 19 : 05 h (see Figure 6.1(a)). External errors, for example due to nowcast uncertainty, are not considered in this setup, as this would falsify the MPTP's reliability. Thunderstorm margins vary between 10000 m at t_0 (resulting in 18.2 % mission area coverage) to 20000 m at $t = T_N = 3600$ s (resulting in 26.4 % mission area coverage). The random variables for MCS are start position, initial course and goal

Table 6.17: *Parameters for the Monte Carlo simulation.*

$\boldsymbol{t_0}$	19 : 05 h
$\boldsymbol{\Delta t}$	50 s
$\boldsymbol{V_T}$	80 m/s
$\boldsymbol{\lvert\Delta\chi\rvert}$	45°
$\boldsymbol{\epsilon}$	0.022°

position, which have to meet the following conditions. Start and goal position are given by uniformly distributed random numbers in the finite latitude/longitude interval of the mission area. In order to be valid, they are not allowed to lie directly on a border or to be covered by thunderstorm, at any time. To ensure, at least to some extent, that the goal is reachable in the nowcast horizon of one hour, the shortest path between the potential start and goal positions is determined in the last nowcast (largest mission area coverage), using visibility and A*-search (see Chapter 4). The length of a shortest path has to be in the range of 30 % to 80 % of the theoretical range of the aircraft, which is given by $V_T 3600s$. The initial course is a random integer number in the interval from $\chi_s \in [0, 359]$. It must not point directly to the goal to avoid trivial (straight-line) trajectories. Thus, in the simplest case a trajectory starts with a turn. Figure 6.7 depicts the distribution of the total trajectory course change for all 9963 successful MPTP trials with SSPH. When start, initial course and goal fulfill all conditions the MPTP is called twice, once with EDH and once with SSPH.

A trial is considered to be successful if the MPTP is able to compute a trajectory within a maximum number of 100 iterations (equal to $EXST$). This strict criterion is particularly well suited for assessing the convergence or rather real-time capability of both motion planning variants, as it is completely independent of hardware and software implementation.

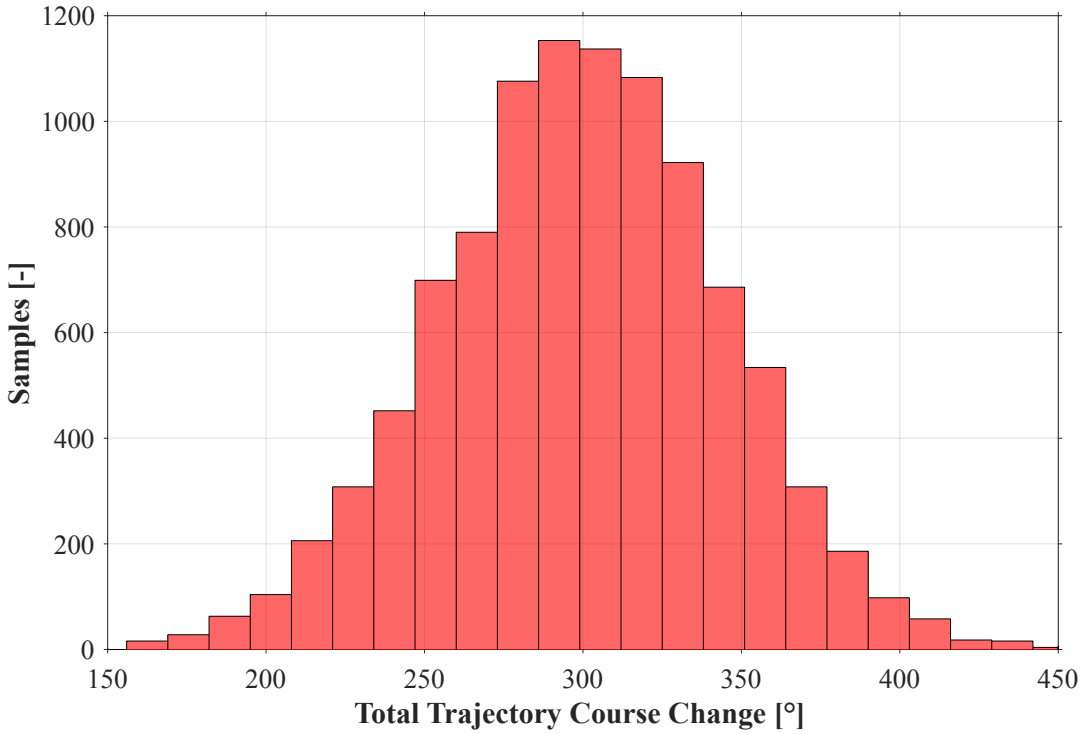


Figure 6.7: *The distribution of the summed course changes of all sampled trajectories shows that due to the initial course constraint no trivial (straight line) trajectories are computed.*

Figure 6.8 shows twelve successive random trajectories of different complexity level, which all converge under 100 iterations. The dotted lines of different color show the final path, while the respective solid lines show the actual trajectory progress. The position of aircraft and obstacles can be observed at six different times. If the MPTP converges, the success of avoidance is guaranteed, as the computed trajectories are free of conflicts.

The MPTP results of each trial are stored, of which success (0 or 1), number of explored states ($EXST$) and number of trajectory states ($TRST$) are of special interest as important performance characteristics can be deduced. The sample mean is calculated by

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_i, \tag{6.2}$$

where n is the number of samples and x is the sample value [125]. According the central limit theorem the resulting distribution is normal as the number of samples goes to infinity. The z-score, which is applicable to normal distribution, is $z_{\alpha/2} = 2.58$ for a 99 % confidence interval CI_{99}. Generally, the confidence interval for a mean sample value, assuming that a sufficiently large number of samples is available (central limit theorem), is calculated by

$$CI = \bar{x} \pm z_{\alpha/2} \frac{\sigma}{\sqrt{n}}, \tag{6.3}$$

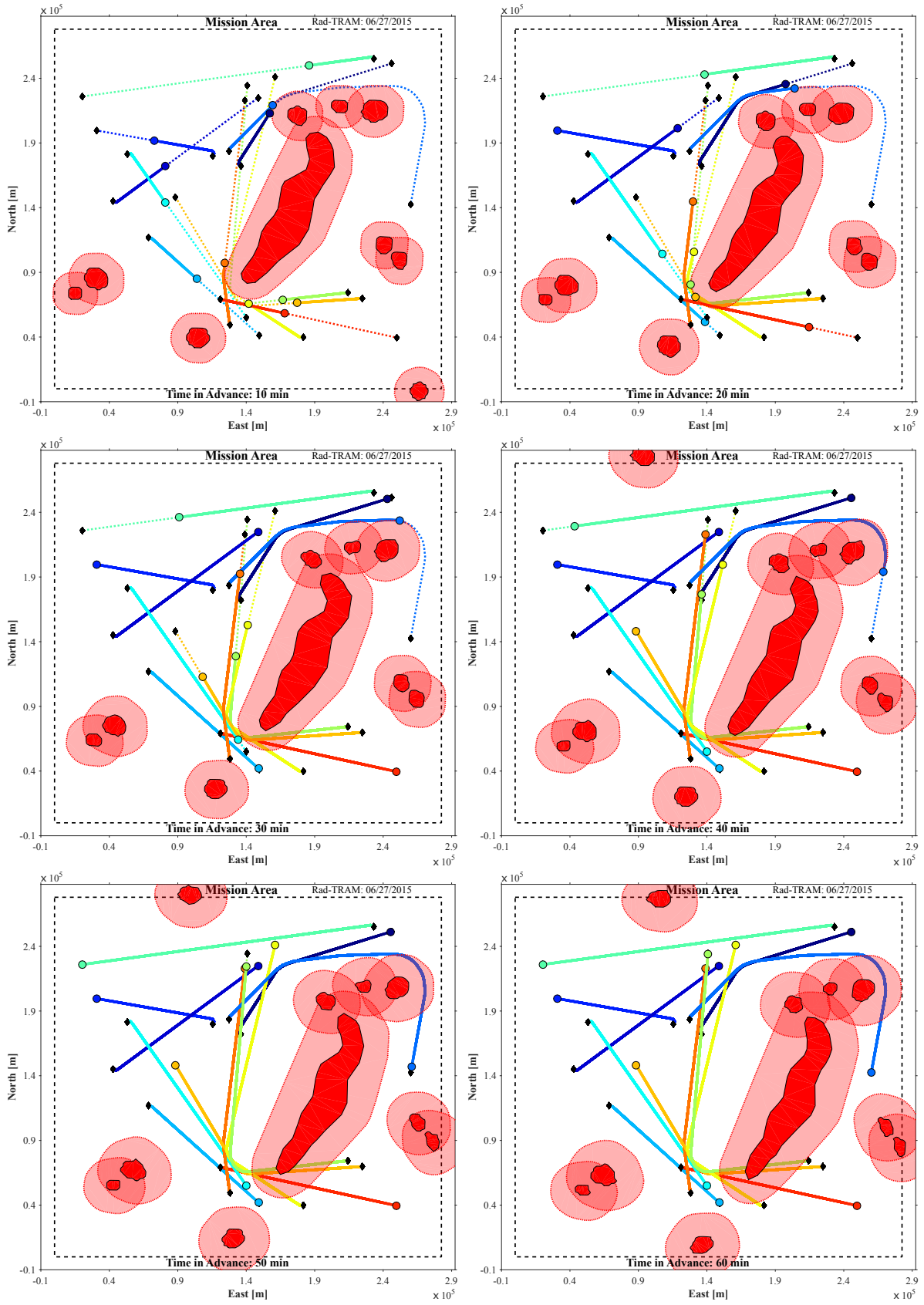


Figure 6.8: *Twelve successive random trajectories for bidirectional turning-flight are depicted, in six time steps with a ten minute interval. The dotted lines show the final path, while the solid lines display the actual covered distance. The black outlined dots, of same color as the trajectories, indicate the actual position of the aircraft.*

where σ is the sample standard deviation [125], which is

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}|x_i - \bar{x}|^2}. \tag{6.4}$$

Figure 6.9 shows the results for mean planning success in bidirectional turning-flight for EDH and SSPH over the number of trials. If the motion planning is successful with both

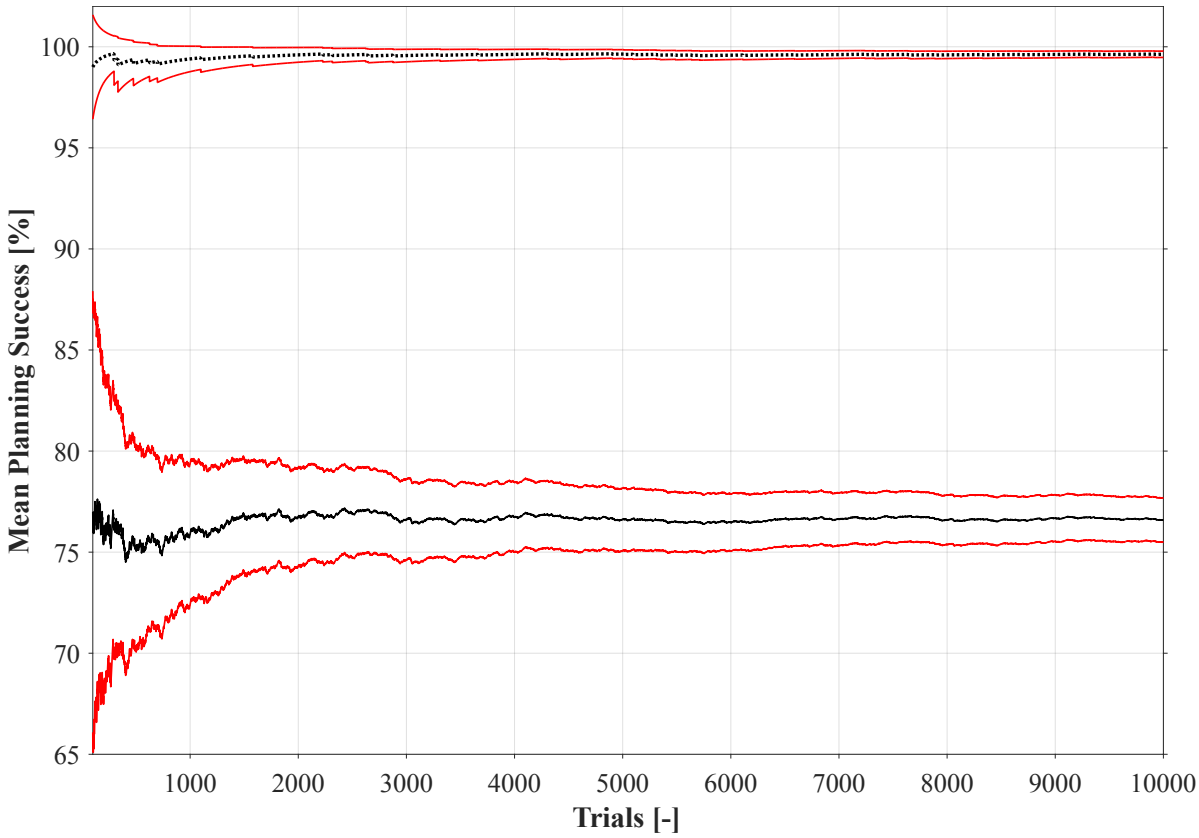


Figure 6.9: *Mean planning success of the MPTP using EDH (solid black line) and SSPH (dotted black line) are depicted in their respective* 99 *% confidence interval (red lines). After* 10000 *trials the confidence interval for SSPH is very narrow ($CI_{99} = [99.47, 99.79]$ %), which indicates that the sample mean is close to the true mean success rate fo the MPTP.*

heuristics (7760 of 10000 trials), the mean trajectory length using EDH is 135966.27 m and 135967.06 m using SSPH, which is a difference of +0.79 m due to the fact that SSPH is inadmissible. Given the large amount uncertainty in the nowcast, the $5.81 \cdot 10^{-04}$ % relative error in length is negligible and shows how little the overall optimality in practical applications is affected by the inadmissibility of SSPH (see Section 4.3.3). The mean ratio between trajectory states and explored states ($\overline{TRST/EXST}$) is a measure for the efficiency of the search. Even in simple scenarios A*-search with EDH is less efficient than with SSPH, which means that more states have to be explored in order to find a feasible trajectory. Figure 6.10 shows the mean search efficiency with both heuristics. The MPTP

has a mean search efficiency of 85.07 % using EDH and 93.74 % with SSPH. However, EDH

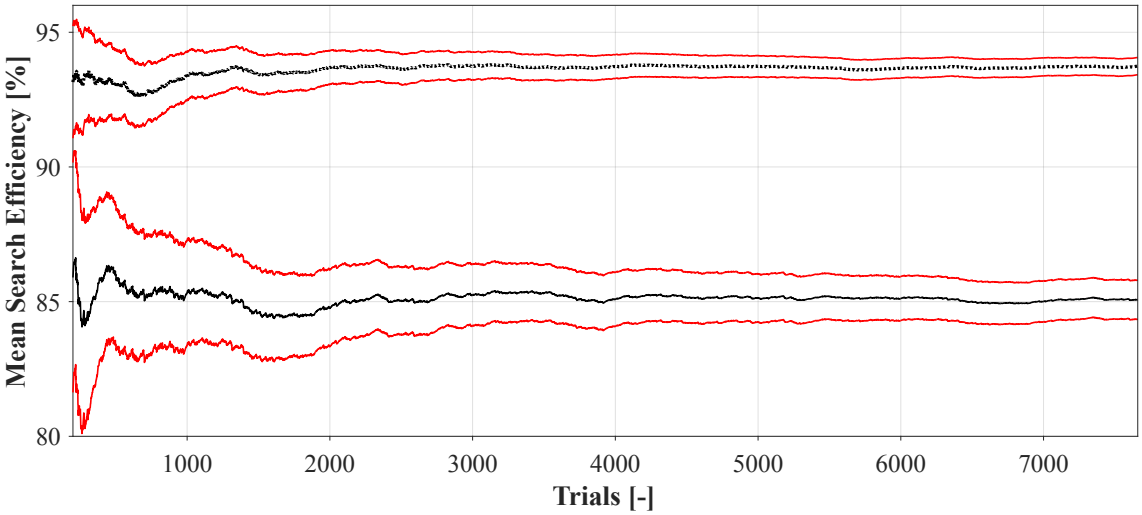


Figure 6.10: *If the MPTP successfully plans trajectories with both heuristics (*7760 *trials): mean search efficiency of the MPTP using EDH (solid black line) and SSPH (dotted black line) are depicted in their respective* 99 *% confidence interval (red lines).*

fails to plan a trajectory in ≤ 100 iterations in 2240 out of 10000 trials. In many cases the number of iterations then exceeds several thousand. Taking the number of resulting trajectory states as a measure for the difficulty of a planning problem, the mean number of trajectory states when EDH finds a solution is $\overline{TRST} = 3.69$ and the mean number of trajectory states when EDH fails and SSPH succeeds is $\overline{TRST} = 10.26$. Figure 6.11 shows the overall mean search efficiency of the MPTP using SSPH in 9963 successful trials. The 99 % confidence interval for the true search efficiency is $CI_{99} = [88.76, 89.59]$ %, which is a very good range. If the maximum iteration limit would be set to 20, the MPTP with

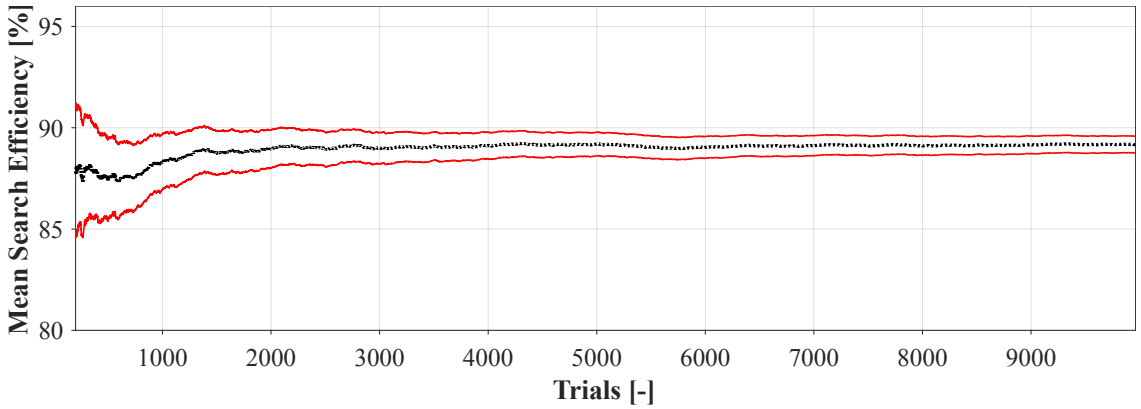


Figure 6.11: *The overall mean search efficiency (*$\overline{TRST/EXST}$*) of the MPTP using SSPH (dotted black line) over* 9963 *successful trials, including* 2203 *trials where EDH failed to converge under* 100 *iterations, is depicted in the* 99 *% confidence interval (red lines). The final mean value is* 89.17 *%, which proves how efficient A*-search with SSPH is.*

SSPH is still able to compute trajectories in 94.79 % of the cases, which can be seen in Figure 6.12. With the same limit MPTP with EDH is only successful in 70.99 %.

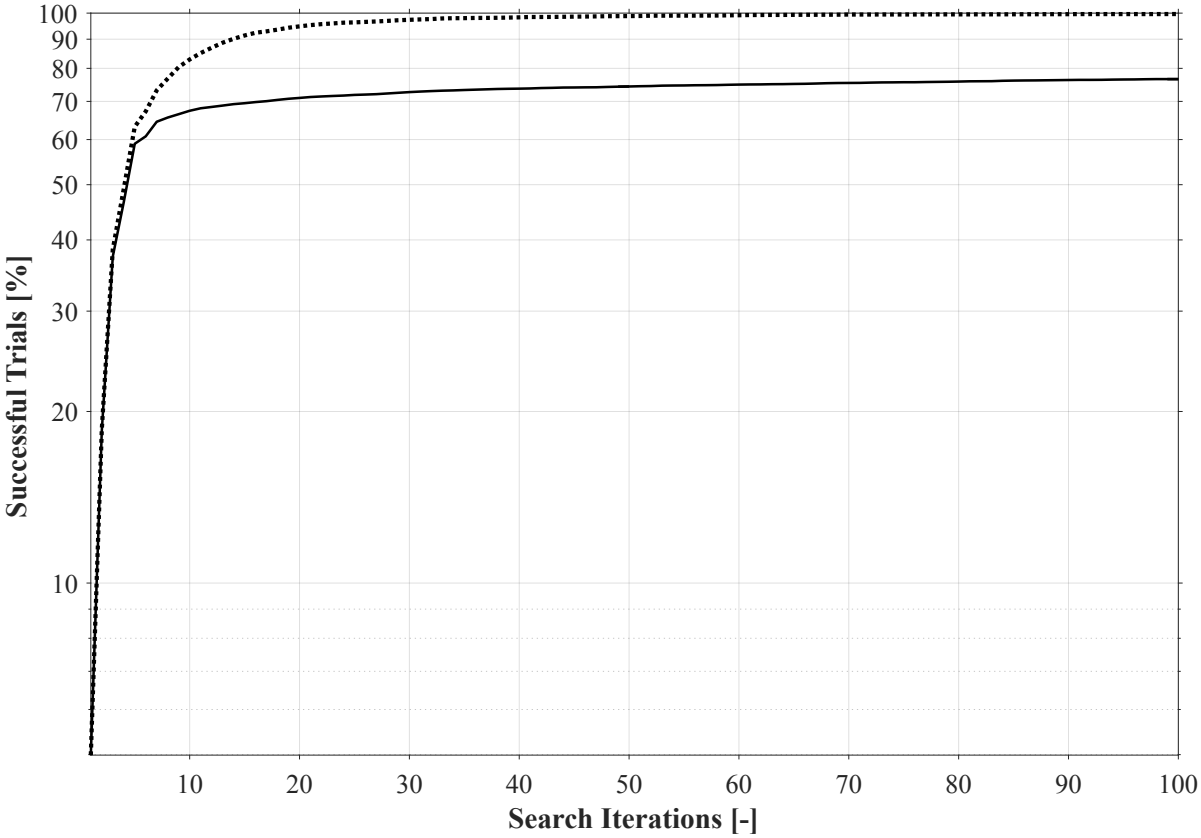


Figure 6.12: *The number of successful trials in the first MCS is plotted against the number of iterations of the MPTP with EDH (solid black line) and SSPH (dotted black line).*

In the following, a second MCS is performed, again with $n = 10000$ independent trajectory samples. In this case the MPTP is only called with SSPH and the number of iterations is not constrained, which would be impracticable with EDH, as it often does not converge. The MCS parameters are the same as in Table 6.17, except for the smaller time increment of the MPTP, which now is $\Delta t = 10$ s. Furthermore, the size of the margins for the selection of admissible start and goal positions is the same as before, however, the size of the actual thunderstorms is slightly smaller and varies between 8500 m at t_0 to 17000 m at $t = T_N = 3600$ s. This has the effect that the start position cannot be directly alongside an obstacle by chance, which in combination with an unfavorable initial course results in an unsolvable problem. Under these conditions, the success rate of the MPTP with SSPH is 100 %, as shown in Figure 6.13. The overall mean search efficiency is 87.51 % (see Figure 6.14), which is only slightly below the 89.17 % of the first MCS (see Figure 6.11). This shows that a time increment variation only has a minor influence

on the search efficiency. The most extensive trial requires 495 iterations, which can be seen in Figure 6.15. The associated trajectory contains 21 states, which indicates that it is a rather intricate trajectory.

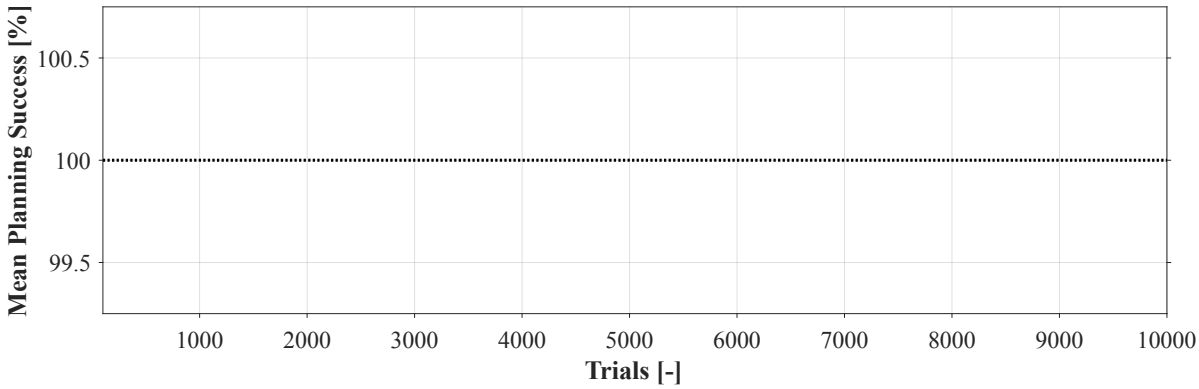


Figure 6.13: *In the second Monte Carlo simulation the number of iterations for each trial is unconstrained and the mean planning success of the MPTP using SSPH (dotted black line) is* 100 *%. Although the informative value of this plot is limited, the author was keen to illustrate this graphically.*

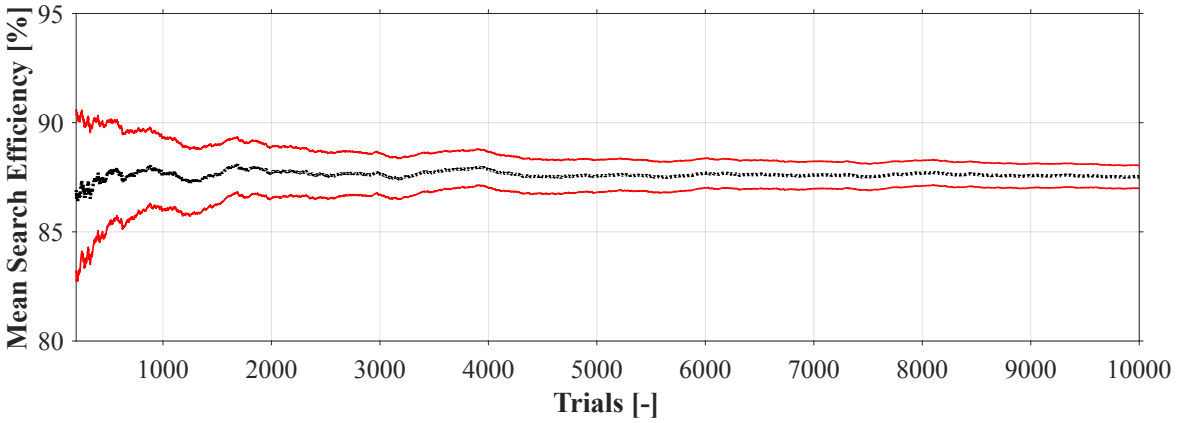


Figure 6.14: *The overall mean search efficiency ($\overline{TRST/EXST}$) of the MPTP using SSPH (dotted black line) is* 87.51 *% in the second Monte Carlo simulation. The* 99 *% confidence interval (red lines) is* $CI_{99} = [86.98, 88.04]$ *%. The search is again very effective, although the time increment is relatively small.*

Regarding a certification, the hazard is that no trajectory can be computed by the MPTP. The severity classification of this failure condition is estimated to be between minor and major, due to the use of generous safety margins and an existing emergency strategy (ESA, see Section 3.2). In this case, the allowable quantitative probability for a class I aircraft (single reciprocating engine and 6000 pounds or less of weight) is between $< 10^{-3}$ and $< 10^{-4}$, according to AC 23.1309-1E [126], which, as demonstrated earlier, can be achieved by the MPTP using SSPH.

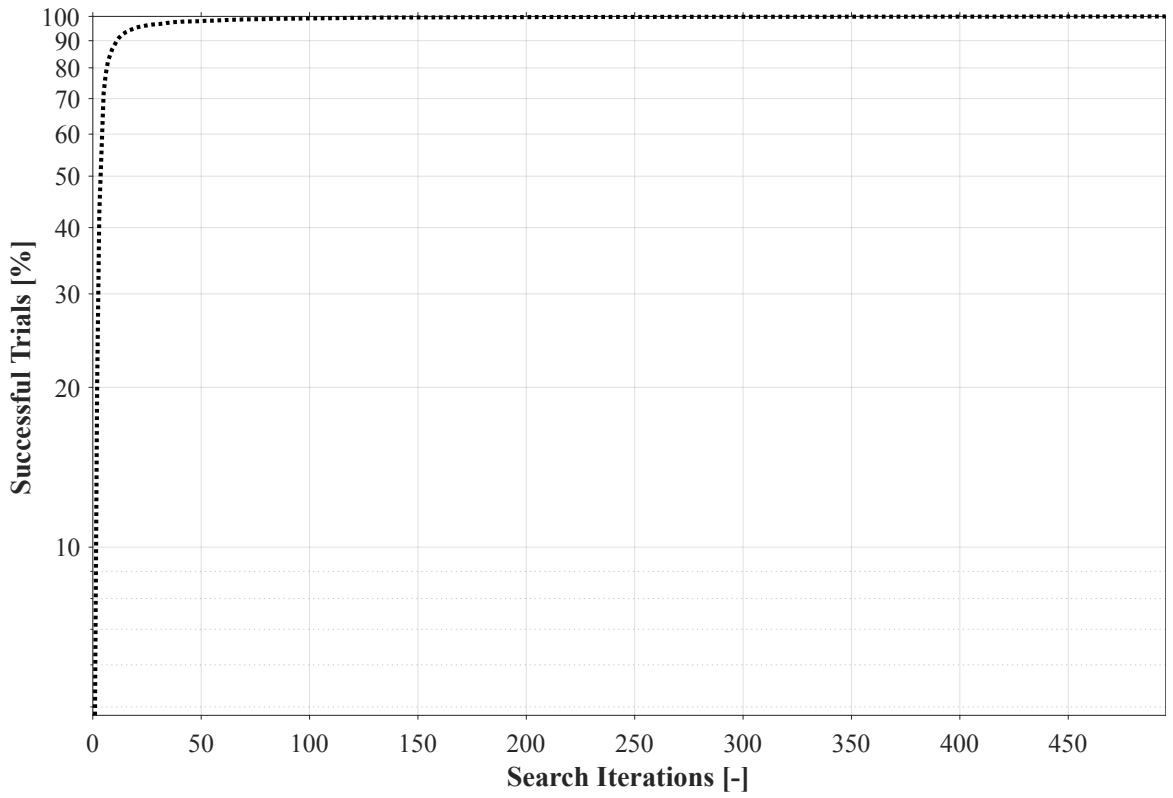


Figure 6.15: *The successful trials of the second Monte Carlo simulation are plotted against the number of corresponding search iterations of the MPTP with SSPH (dotted black line). The maximum number of necessary iterations is* 495*. About* 98 *% of the trials can be accomplished with* ≤ 50 *explored states.*

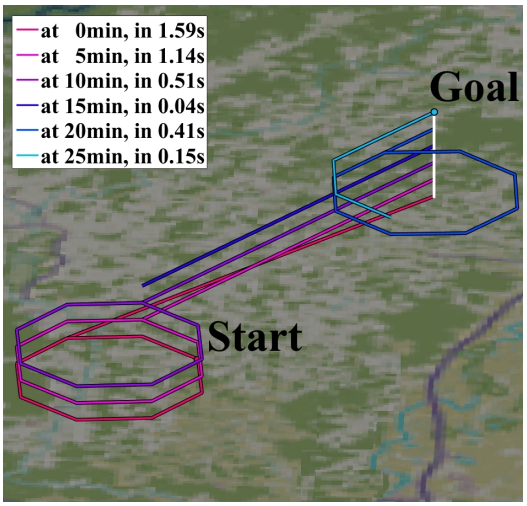
6.3 Automatic Holding Pattern Scenario

Table 6.18: *Parameters for the automatic holding Scenario. Index s stands for start state and index g for the goal state.*

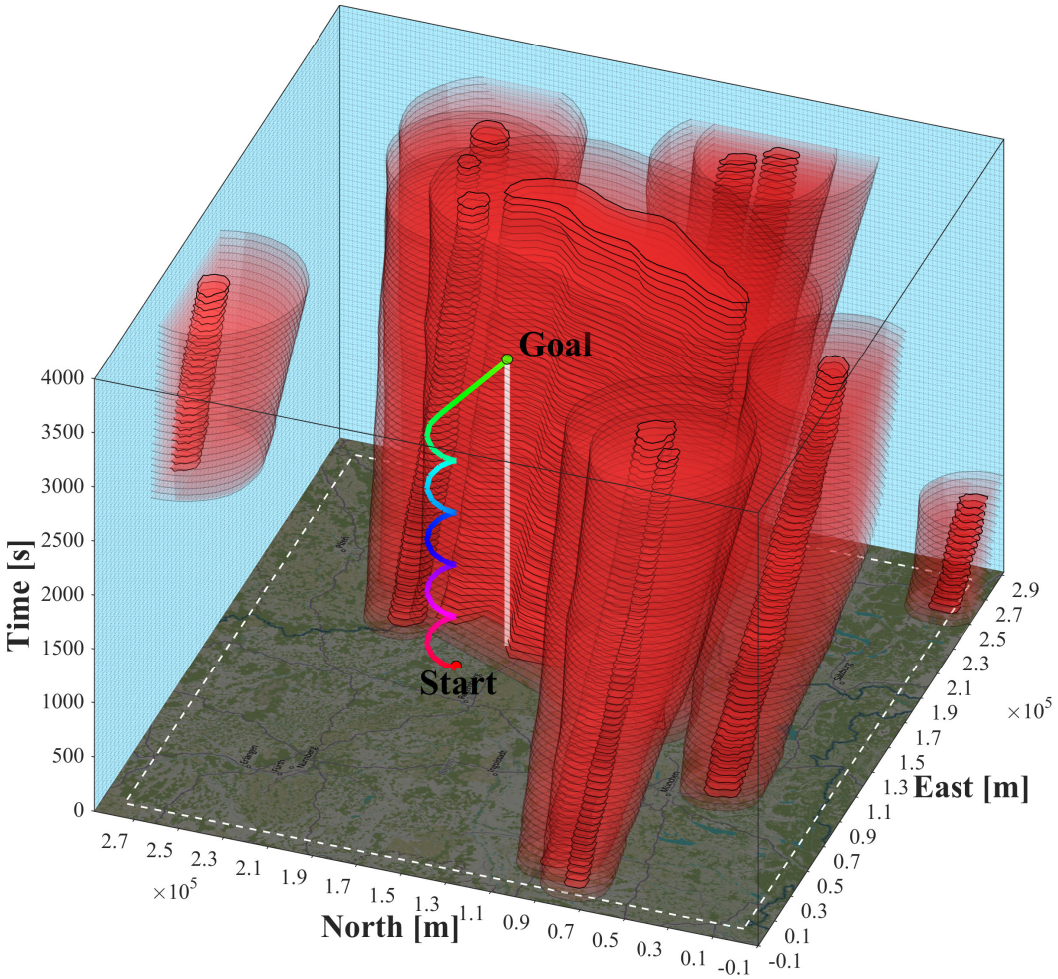
$\boldsymbol{\varphi_s}$	49.134°
$\boldsymbol{\lambda_s}$	12.168°
$\boldsymbol{\chi_s}$	220°
$\boldsymbol{\varphi_g}$	49.006°
$\boldsymbol{\lambda_g}$	11.980°
$\boldsymbol{t_0}$	19 : 05 h
$\boldsymbol{\Delta t}$	60 s
$\boldsymbol{V_T}$	80 m/s
$\boldsymbol{\lvert\Delta\chi\rvert}$	45°
$\boldsymbol{\epsilon}$	0.022°

Table 6.18 lists the parameters for the automatic holding pattern example. The Figure 6.16(a) shows, how the MPTP plans holding patterns and adapts to changing nowcasts by replanning a trajectory in every iteration. In this example A*-search with EDH is applied. The meteorological data is the same, as in the previous section. In the first iteration

of the MPTP, at $t_0 = 19:05$ h UTC, the algorithm plans four and a half circular holdings before ending the hold at $t_0 + 38$ min and arriving scheduled at the just uncovered goal at $t_0 + 44$ min$= 19:49$ h (see Figure 6.16b). In the second iteration at $t_1 = t_0 + 5$ min, the updated nowcast allows, that the aircraft ends the hold after three circles at $t_1 + 24$ min to arrive scheduled at the goal $t_1 + 31$ min. In the third iteration at $t_2 = t_0 + 10$ min the nowcast indicates, that the aircraft can exit the hold after one and half circles at $t_2 + 12$ min to arrive at the goal at $t_2 + 18$ min. In the fourth iteration $t_3 = t_0 + 15$ min the algorithm plans to exit the hold immediately with an expected time of arrival of $t_3 + 5$ min. At $t_4 = t_0 + 20$ min the margins around the nowcast cover the goal forcing the algorithm to plan an additional hold. Finally at $t_5 = t_0 + 25$ min the aircraft heads to the uncovered goal. The final flight duration is $t_5 + 4$ min$= 29$ min, and the actual time of arrival is $19:34$ h.



(**a**) *All consecutive MPTP trajectories.*



(**b**) *Three dimensional illustration of the first MPTP trajectory.*

Figure 6.16: *(a) Anticipatory trajectories including automatic holding patterns with runtimes for the MPTP iterations in the legend. The trajectories are depicted one above the other to illustrate the different points in time, at which they are computed. As the method from Section 5.4 is applied, a holding pattern is represented by a regular octagon. (b) First iteration of the MPTP at t_0 in three dimensions. The static goal is indicated by a white line. As soon as the goal is uncovered, it is approached.*

6.4 Moving Goal Scenario

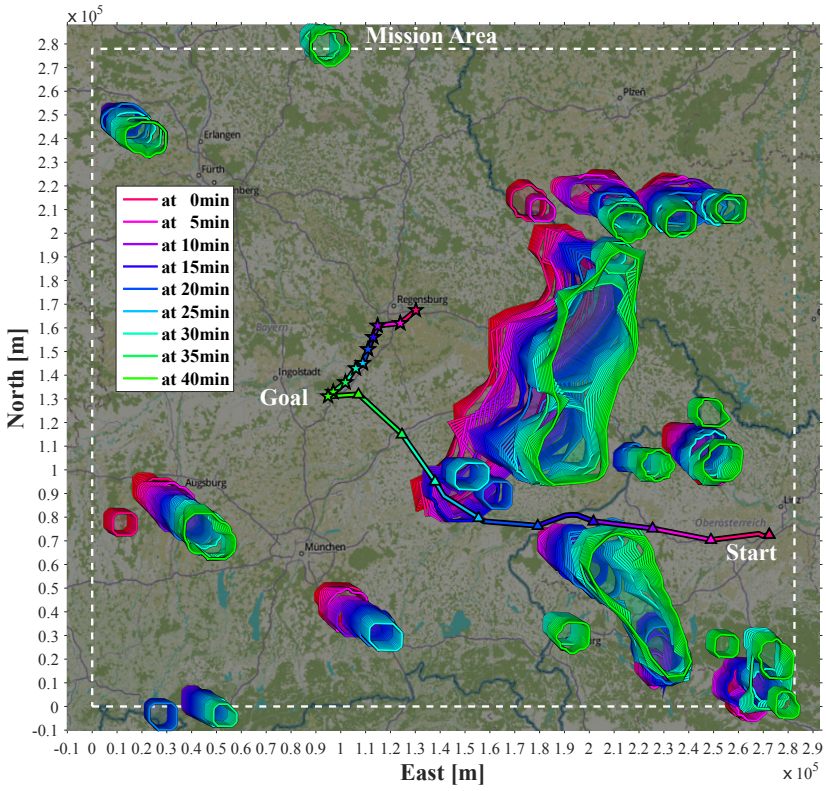
Table 6.19: *Parameters for the moving goal Scenario. Index s stands for start state and index g for the goal state.*

$\boldsymbol{\varphi_s}$	48.154°
$\boldsymbol{\lambda_s}$	13.868°
$\boldsymbol{\chi_s}$	205°
$\boldsymbol{t_0}$	19 : 05 h
$\boldsymbol{\Delta t}$	30 s
$\boldsymbol{V_T}$	80 m/s
$\boldsymbol{\lvert\Delta\chi\rvert}$	45°
$\boldsymbol{\epsilon}$	0.022°

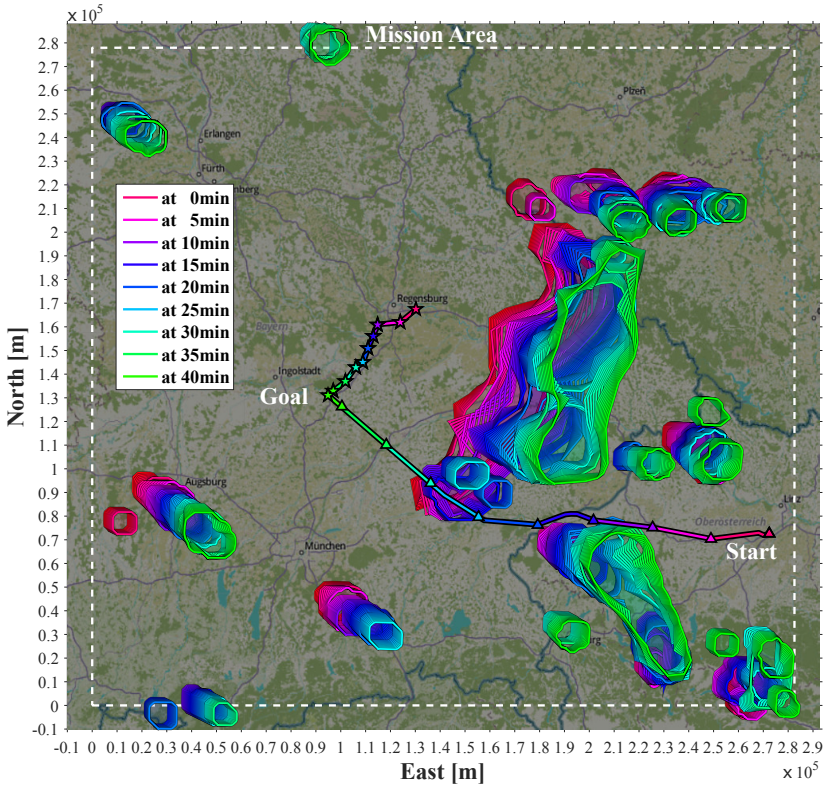
Table 6.19 lists the parameters for the moving goal examples. The applied heuristic is the Euclidean distance. In Figure 6.17(a) the actual position of the moving goal (colored star markers) is updated with every nowcast, which results in a reactive navigation. In Figure 6.17(b) the MPTP has information about the future goal states in advance (see Table 6.20), which allows a tactical trajectory directly towards the estimated rendezvous location. Table 6.20 shows the states of the moving goal x_g used in this example.

Table 6.20: *The goal state consists of coordinates parameterized by time.*

Time [min]	$\boldsymbol{\varphi_g}$ [°]	$\boldsymbol{\lambda_g}$ [°]
0	49.006	11.980
5	48.957	11.896
10	48.948	11.776
15	48.903	11.750
20	48.858	11.723
25	48.804	11.697
30	48.786	11.657
35	48.732	11.604
40	48.696	11.537
45	48.678	11.511
50	48.651	11.484
55	48.606	11.471
60	48.597	11.405



(**a**) *Reactive trajectory to the goal state.*



(**b**) *Anticipatory trajectory to the goal state.*

Figure 6.17: *(a) The information about the position of the goal state is updated simultaneously with the nowcast. This leads to reactive planning and thus results in a pursuit curve, which becomes apparent approximately from the 28th minute (first bend after the light blue triangle). In (b) a prognosis of the goal movement is available, which results in a straightway trajectory towards the moving goal.*

Chapter 7

Discussion

This chapter discusses the simulation results from the previous Chapter 6, with which Objective 5 is accomplished. Comparing Figure 6.1(a) and 6.1(b) illustrates the considerable amount of prediction uncertainty the MPTP is subjected to. This manifests in the differences, between the anticipatory trajectories (monochrome lines) in Figures 6.3 to 6.5. Although solely relying on external information by nowcast, the MPTP safely avoids thunderstorms in all presented scenarios (Section 6.1 to 6.4). The key is the implicit consideration of uncertainty, by replanning a near-optimal anticipatory trajectory for the expected case, with every update of the nowcast.

Figure 6.2 illustrates that trajectory planning is done in three-dimensions as time has to be considered in dynamic environments. The anticipatory nature of the trajectory is shown by the absence of pursuit curves. It is free of conflicts and the vertical slope is always positive, which means that time is monotonically increasing. Due to the constant ground speed, the vertical slope is likewise constant.

The final trajectories, from start state to goal, in the first and second scenario, are assembled by the first legs of the anticipatory trajectories, with a duration of ΔT_N. As each of them is the best guess at the time, in both scenarios, the final trajectory is shorter, than expected in the first MPTP iteration. Probabilistic margins help to prevent the MPTP from replanning substantially diverging trajectories.

In the second scenario, the goal lies in the direction of the moving thunderstorms (Figure 6.5). At 19:05 h, the thunderstorms are moving at ground speeds ranging from 6.2 m/s to 17.6 m/s and courses ranging from 88.5° to 128.5° (mean 110°). The beeline distance to the goal is 170.9 km and the course is 303.7°, which is almost opposite to the thunderstorms. Although uncertainty is explicitly considered, in the fourth MPTP iteration, the aircraft is suddenly inside a safety margin, but still outside the cell itself

(fourth, dark blue triangle from the start). A robust feature is the addition of the aircraft to X_{free}, if the aircraft is suddenly trapped. Thus, the algorithm is able to exit the conflict as fast as possible, while only marginally changing the course. This behavior is compliant with FAA rules [80]. Due to the fast development of thunderstorms, the nowcast uncertainty is sometimes so large, that even extended probabilistic safety margins cannot completely rule out such incidents, without excessive blockage of airspace. The only way to prevent this kind of unexpected incident, is the usage of an onboard radar.

The results in Section 6.1 show, that trajectory planning with unidirectional constrained turning is faster for all heuristics. This is intuitive, as only one instead of two auxiliary states (Section 5) is added into the open list, which reduces the branching factor for the A*-search. It is important to keep the number of additional states in the open list, as small as possible. However, although not evident from the presented results, the unidirectional constrained flight can lead to suboptimal trajectories, due to unnecessary turning and even compromises convergence in some cases.

The performance of A*-search with three different heuristics is compared in Section 4.3. The ratio between the number of trajectory states (TRST) and the number of explored states (EXST), in order to compute the trajectory, is taken as criterion for effectiveness of the search. If the heuristic cost H is set to zero, A* basically becomes Dijkstra's algorithm. As expected this results in the lowest ratios for TRST/EXST, in all iterations of both scenarios. The reason is, that the priority queue basically organizes the candidate states in ascending order, regarding their G-cost. The uninformed search leads to the exploration of unnecessary candidate states. When bidirectional turning flight is applied, Dijkstra does not converge in hours, which is why in both scenarios no data is presented. As two newly added auxiliary states are near the actually explored state, they are high in the priority queue. This inhibits the progression of the search towards the goal. In the first scenario, the TRST/EXST hit ratio, for unidirectional turning, is ≤ 2.2 %, when obstacles are in between start and goal (Table 6.2). In the second scenario, the maximum allowed deviation is larger than in the first scenario, which causes more candidate states in the open list. The TRST/EXST ratio, in the first four MPTP iterations, is extremely poor (Table 6.10). In the third MPTP iteration, the number of explored states is 15830 times higher, than that of A*-search with SSPH (Table 6.13). These results indicate, that Dijkstra is unsuitable for the present application in state space.

The A* performs better with EDH, as the search is informed. Without obstacles in between start and goal, the EXST/EXST ratio with EDH is en par with SSPH, in uni- and bidirectional flight (last MPTP iterations in Tables 6.5, 6.8, 6.13, 6.16). Although

the pruning of nontangent edges prevents unnecessary candidate states, the search with EDH can be rather slow, especially in crowded environments. The reason is, that the Euclidean distance, from start to goal, ignores obstacles. In unidirectional turning flight, the unnecessary exploration of auxiliary states is inhibited by increasing values for Euclidean distance when performing a turn. This is why A* with EDH performs reasonably, in both scenarios for unidirectional turning. In the first scenario (Table 6.5), A* with EDH explores at most 4.46 times more states, than A* with SSPH and in the second scenario, at most 38.5 times more (Table 6.13). However, by adding left and right auxiliary states in every A* iteration, the inhibition of excessive exploration is abolished and the performance is unacceptable, for the first MPTP iterations in the Tables 6.8 and 6.16. The search is unwilling to increase the H-cost, in order to fly around obstacles, and mainly explores the auxiliary states in front of the obstacles.

The A*-search with SSPH has the highest TRST/EXST hit rate in all MPTP iterations of both scenarios, as obstacles are taken implicitly into account. As only the most promising states are explored, the performance between uni- and bidirectional turning flight is almost identical. Only in the first scenario, in the first MPTP iteration, the bidirectional search explores two states more than in the unidirectional case (compare Table 6.4 and 6.7). In the second scenario, the number of EXST is identical for all MPTP iterations (compare Table 6.12 and 6.15). The introduced SSPH is not admissible as overestimation of the heuristic cost is theoretically possible, which is why the optimality of trajectories cannot be guaranteed. This is due to the fact, that SSPH is computed in the free estimated state space from the perspective of the actual initial state $X_{free}(\mathrm{x}_{in})$. Strictly speaking, the shape of the obstacles is only valid, when staying on the radials outgoing from the initial state. As shortest paths for the adjacent states, to the initial state, are evaluated around approximate obstacle shapes, under- as well as overestimation of the H-cost is possible. This mainly depends on the motion of obstacles relative to the shortest static path. If it passes on the side of movement direction, the H-cost is underestimated. Underestimation slows the convergence, yet leads to an optimal result and is therefore admissible. If the shortest path is on the backside of obstacle movement, the H-cost is overestimated, which can lead to suboptimal results. An overestimation of the H-cost is sometimes done on purpose, using an inflation factor $\epsilon \geq 1$, as this can speed up the search [127, 106]. However, the SSPH generally tends to underestimate the cost-to-go, due to nonnormal shortest path segments, in the isochronous rings from the initial state (Figure 4.16).

In the first scenario, in the fourth iteration of the MPTP, the computed trajectory

with SSPH (Table 6.4) is 200 m longer than with Dijkstra (Table 6.2) or A* with EDH (Table 6.3). Generally, if there is a difference between the trajectories, it is small. Compared to the large amount of nowcast uncertainty and extensive margins, these differences are negligible. The fact, that Dijkstra and A*-search with EDH produce identical trajectories in 19 of the 20 presented anticipatory trajectories, indicates, that A* with SSPH frequently produces near-optimal results. Overall, the A*-search with SSPH is well-suited for planning in dynamic environment. It is considerably faster and more consistent, compared to Dijkstra and A* with EDH. The runtime for trajectory planning is crucial for an online application. For example, in Chapter 6, the aircraft travels at $V_T = V_G = 80$ m/s. In the first MPTP iteration, of the second scenario, the aircraft covers a distance of 80 m/s·0.42 s=29.6 m, using A* with SSPH, whereas it covers 80 m/s·11.29 s= 903.2 m, using A* with EDH, while the trajectory is computed.

In summary, in Section 6.1 the Objectives 1, 3 and 4 are accomplished. It is demonstrated that the presented motion planning algorithm is capable to successfully avoid time-varying thunderstorms in anticipatory fashion, based solely on uncertain external environmental prediction, while staying inside an assigned mission area. Furthermore, the example in Figure 6.4 demonstrates that the MPTP is capable of handling wind. Due to implicit consideration, using approximate future states for constant wind (see Figure 3.8), the MPTP computes a series of trajectories that get the aircraft to the goal at almost the same time as in the wind-free case (see Figure 6.3), despite the initial headwind.

The results in Section 6.2 show the potential of the MPTP, which is able to compute feasible trajectories in a real scenario quickly and extremely reliably, which fulfills Objective 2. The MPTP with SSPH, which achieves a mean success rate of over 99 % in the iteration constrained Monte Carlo simulation, meets the demand for a simple and reliable motion planning algorithm from Chapter 2. Without the limitation to 100 iterations the success rate of EDH is higher, however, at the same time the number of mean explored states increases considerably, making it too slow for an online application. Another finding is that the trajectories computed with SSPH are very close to those of EDH, regarding optimality. The mean relative length error, compared to the optimal trajectories computed with EDH, is only $+5.81 \cdot 10^{-4}$ %, which is negligible considering how much faster and more reliable SSPH is. Furthermore, in real operations the resulting errors caused by uncertainty, for example in the environmental prediction, are orders of magnitude larger. The mean success rate in the second Monte Carlo simulation is 100 % for an unconstrained number of iterations. The highest necessary number of explored

states is 495, when applying a relatively small time increment of $\Delta t = 10s$. Interestingly the mean search efficiency is barely affected by the time increment.

Finally, it should be noted that it is not possible to draw a general statement regarding the mean performance of the MPTP based on the performed Monte Carlo simulation, as the values determined are only valid for the present setups. To determine more reliable values, testing has to be carried out in a large number of different scenarios, which is at least in the order of magnitude of the applied 10000 trials. Due to the large number of trials the execution of this experiment is planned as future work. The aim is to develop a counter-optimization, which subjects the MPTP to a kind of reactive stress test.

As it may not be possible to find a complete trajectory in an allotted time interval an additional partial planning strategy is needed, in order to avoid passivity [31] and guarantee decisioning. A simple and safe procedure is to continue straight flight inside an estimated safe area (ESA), as presented in Section 3.3. Another strategy is to perform holdings in free state space. The example in Section 6.3 shows the ability of the planner, to automatically plan holding patterns, e.g. if the goal is temporarily covered. This is important, as bounded margins (see Section 3.1) sometimes conservatively cover free space, including the goal state, which reduces the probability of finding a solution. Like before, the final trajectory takes less time, than first guessed. Due to periodic replanning (Chapter 2) and estimation of future aircraft states (Section 3.2), the planner also has the innate ability to deal with a time-varying goal state, which is demonstrated in Section 6.4. If a prognosis is available, the MPTP finds the shortest trajectory to the moving goal, through time-varying thunderstorms (see Figure 6.17(b)).

Chapter 8

Conclusion and Outlook

8.1 Conclusion

A viable method for robust trajectory planning in uncertain dynamic environments was presented. It enables anticipatory avoidance of static and moving obstacles. The shape and movement of the polygonal obstacles can be arbitrary. The combinatorial algorithm is resolution-complete and near-time-optimal for the expected case, under the assumption of constant true airspeed and angle of climb. Since it always performs the same deterministic computations, the results are repeatable, which is advantageous regarding a certification. Also favorable, is the fast and highly reliable computation of trajectories in combination with the novel shortest static path heuristic, as demonstrated in Section 6.2. Due to anticipatory planning and consideration of nonholonomic turning-flight constraint of fixed-wing aircraft, computed trajectories do not necessarily require postprocessing, which was demonstrated in [3]. The ability to automatically plan holding patterns, if the goal itself or the access is covered by obstacles, further improves the probability for success. The capability, to plan with a time-varying goal state, expands the horizon of possibilities. In conclusion, it can be stated that the presented MPTP fulfills all the objectives set out in Chapter 1 and is ideally suited for online motion planning in dynamic environments. The research contributions of this thesis and respective results are summarized in the following.

- Contribution 1: In Section 3.3, future estimated state space and estimated future conflict areas were introduced. Time-varying thunderstorms are transformed to state space obstacles by superposition of their prediction (including uncertainty) with contemporaneous estimated future aircraft states (from an initial state). A set of states, in which the aircraft is estimated to be in collision with thunderstorms,

is called estimated conflict area (ECA). The free estimated state space (EX_{free}) is the difference of a workspace W (mission area) and the set of ECAs. The presented concept enables fast and reliable anticipatory trajectory planning in dynamic environments by solving geometric problems, which was demonstrated in numerous simulations in Chapter 6.

- Contribution 2: The determination of the visibility between discrete obstacles can be a bottle neck, regarding a fast computation of trajectories. In Chapter 4, several methods to minimize the computational load were introduced. Line-smoothing of polygonal obstacles is one of them (Section 4.1.1). Although this results in some inaccuracy, it is minor regarding the considerable uncertainty in the nowcast. Thereby, the safety of trajectories is not affected, as it is applied to margins around the actual thunderstorms. In Section 4.1.2, a novel method to determine tangent edges and bitangent edges was introduced, which exclusively uses matrix operations. The stored trigonometric information can be theoretically used to determine the visibility of the filtered edges by counting the number of intersections with the edges of EX_{free}, taking geometric degeneracies into account. The use of exclusively tangent/bitangent edges in a visibility graph, reduces the mean branching factor, which was analyzed at the end of Section 4.1.2. Compared to a grid representation of the state space the proposed visibility methods contribute to a more efficient search, which is the basis for real-time motion planning in dynamic environments.
- Contribution 3: A simultaneous construction and search of a tangent visibility graph to the goal, is presented. With this method the shortest path (static obstacles, see Section 4.1.2) or trajectory (moving obstacles, see Section 6.1 and 6.3) can be efficiently computed, which is explained in the Sections 4.1.2, 4.3.1 and 4.3.2. The time complexity is the same, as for the continuous Dijkstra, with $O(|V_{EX}| \log |V_{EX}|)$, where $|V_{EX}|$ is the number of vertices in the estimated state space. The search is complete and optimal, if a consistent heuristic function is applied. Using A*-search with Euclidean distance heuristic in a dynamic environment, is considerably faster at building a partially shortest trajectory map, than Dijkstra's algorithm, which was demonstrated in Chapter 6.
- Contribution 4: Section 4.2 features an A*-search, to compute near-optimal trajectories in dynamic environments. By using the concept of estimated state space and partial shortest trajectory map, near-optimal trajectories in dynamic environments can be computed using different heuristic functions. However, the convergence can

be rather slow, when using common heuristic functions, like the Euclidean distance. In most cases, the convergence can be significantly improved with the help of the novel SSPH heuristic function, which is demonstrated in Chapter 6, by comparison with Dijkstra's algorithm and the Euclidean distance. The omission of certain queries from the original A*-search was proposed in Section 4.2.2, for the special case, in which all obstacles are moving. This measure can improve the runtime, at a low risk of sacrificing optimality.

- Contribution 5: The novel shortest static path heuristic (SSPH) for an efficient A*-search in dynamic environments was introduced in Section 4.3.3. It is a particularly targeted cost-to-go estimator, which enables fast combinatorial planning in state space X. This opens up the possibility, to use the presented MPTP online. Although, the presented heuristic is inadmissible and optimality cannot be guaranteed, as an overestimation of the cost-to-go is possible, the results in Section 6.2 indicate, that the computed trajectories are generally identical to optimal methods, for example A*-search using Euclidean distance heuristic.
- Contribution 6: The presented planner is able to consider nonholonomic constraints of a fixed-wing aircraft by adding feasible states to the search-graph in Chapter 5. For their determination different methods, ranging from low- to high-fidelity representation of aircraft dynamics, were introduced. As states are waypoints parameterized by time, it is important to asses their feasibility. The method presented in Section 5.1 is a fast and simple way to evaluate both feasibility of fly-by and fly-over states. For an improved guidance of low-performance aircraft, feasible states can be simulated by a fast noniterative method, which is based on the Taylor series and analytically computes reinitialization times in order not to exceed the commanded course change or maximum bank angle (see Section 5.3.6). Additionally, for the case in which the maximum bank angle is not achieved before the commanded course change, a method was introduced, which improves the accuracy of the TS approximation by tuning the piecewise constant roll-in and roll-out rates.
- Contribution 7: The discrete representation of obstacles and their uncertainty can lead to an excessive coverage of free space. The presented trajectory planner has the immanent ability to plan holding patterns, which was described in Section 5.4. This is especially useful, if the goal state is covered or the access is blocked by thunderstorms. It is achieved by constraining the turning-sense of the aircraft, which results in circular patterns, which are an approved maneuver for unmanned

aircraft. The ability to hold increases the likelihood to compute a sensible and intuitive trajectory, which can be seen in Section 6.3.

8.2 Outlook

Motion planning in time-varying environments is a very exciting scientific field. As for the presented planner, there is still potential for improvement and open questions. Through optimization of the algorithm and the implementation in C/C++, a significant improvement in the runtimes of the MPTP can be expected.

In order to obtain reliable statements regarding the performance of the MPTP a large number of experiments in different scenarios has to be carried out, which can be done using the Monte Carlo simulation presented in Section 6.2. In this context, another interesting point to investigate is the influence of margin scaling regarding the length of resulting trajectories and the convergence of the MPTP, for which a stochastic analysis is ideally suited.

The planner is sensitive to certain parameters, which influence the computational time and quality of the trajectories, e.g. the time step for the prediction and optimization. An automated selection of these parameters can improve the results.

Until now the algorithm plans with at least piecewise constant velocity. It is planned to include variable velocity into the MPTP, as parameter for the optimization.

Provision is made to offer the possibility to climb during holding phases. Especially for aircraft like HAPS, with little energy on board, it is important to use it as effectively as possible. In order to plan safe trajectories in three spatial dimensions, the concept of estimated conflict surfaces (see Section 3.3) will be applied.

The coupling of MPTP and gradient based optimization, e.g. the optimal control tool FALCON.m by TUM-FSD [128], is especially interesting. The model predictive trajectory planner computes a trajectory in a dynamic environment, which is used as initial guess for the optimization of the trajectory of a simple point-mass model. The optimum trajectory of the simple model is then used for the optimization of the trajectory of a high-fidelity model, e.g. 6-DoF aircraft simulation. By comparison with optimized trajectories, based on other initialization methods, e.g. rapidly-exploring random trees [42, 41], the capability of the MPTP as initial guess generator for dynamic environments can be examined.

Bibliography

[1] R. K. Jenamani and A. Kumar, "Bad weather and aircraft accidents–global vis-à-vis indian scenario," *Current Science*, pp. 316–325, 2013.

[2] A. J. Fultz and W. S. Ashley, "Fatal weather-related general aviation accidents in the united states," *Physical Geography*, vol. 37, no. 5, pp. 291–312, 2016.

[3] R. Müller, J. J. Kiam, and F. Mothes, "Multiphysical simulation of a semi-autonomous solar powered high altitude pseudo-satellite," in *2018 IEEE Aerospace Conference*, pp. 1–16, IEEE, 2018.

[4] G. Li and S. P. Baker, "Crash risk in general aviation," *Jama*, vol. 297, no. 14, pp. 1596–1598, 2007.

[5] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information.," *Psychological review*, vol. 63, no. 2, p. 81, 1956.

[6] J. Canny, B. Donald, J. Reif, and P. Xavier, *On the complexity of kinodynamic planning*. IEEE, 1988.

[7] B. Donald, P. Xavier, J. Canny, and J. Reif, "Kinodynamic motion planning," *Journal of the ACM (JACM)*, vol. 40, no. 5, pp. 1048–1066, 1993.

[8] J. Reif and M. Sharir, "Motion planning in the presence of moving obstacles," *Journal of the ACM (JACM)*, vol. 41, no. 4, pp. 764–790, 1994.

[9] J. S. Mitchell *et al.*, "Geometric shortest paths and network optimization," *Handbook of computational geometry*, vol. 334, pp. 633–702, 2000.

[10] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[11] A. Nilim, L. El Ghaoui, V. Duong, and M. Hansen, "Trajectory-based air traffic management (TB-ATM) under weather uncertainty," in *Proc. of the Fourth International Air Traffic Management R&D Seminar ATM*, 2001.

[12] J. Krozel, S. Penny, J. Prete, and J. Mitchell, "Comparison of algorithms for synthesizing weather avoidance routes in transition airspace," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, p. 4790, 2004.

[13] J. Krozel, C. Lee, and J. S. Mitchell, "Turn-constrained route planning for avoiding hazardous weather," *Air Traffic Control Quarterly*, vol. 14, no. 2, pp. 159–182, 2006.

[14] J. Pannequin, A. Bayen, I. Mitchell, H. Chung, and S. Sastry, "Multiple aircraft deconflicted path planning with weather avoidance constraints," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, p. 6588, 2007.

[15] M. Kamgarpour, V. Dadok, and C. Tomlin, "Trajectory generation for aircraft subject to dynamic weather uncertainty," in *49th IEEE Conference on Decision and Control (CDC)*, pp. 2063–2068, IEEE, 2010.

[16] D. Hentzen, M. Kamgarpour, M. Soler, and D. Gonzalez-Arribas, "On maximizing safety in stochastic aircraft trajectory planning with uncertain thunderstorm development," *Aerospace Science and Technology*, vol. 79, pp. 543–553, 2018.

[17] S. Summers, M. Kamgarpour, J. Lygeros, and C. Tomlin, "A stochastic reach-avoid problem with random obstacles," in *Proceedings of the 14th international conference on Hybrid systems: computation and control*, pp. 251–260, ACM, 2011.

[18] R. Bellman, *Dynamic Programming*. Princeton, NJ, USA: Princeton University Press, 1957.

[19] X. Yu and Y. Zhang, "Sense and avoid technologies with applications to unmanned aircraft systems: Review and prospects," *Progress in Aerospace Sciences*, vol. 74, pp. 152–166, 2015.

[20] M. Skowron, W. Chmielowiec, K. Glowacka, M. Krupa, and A. Srebro, "Sense and avoid for small unmanned aircraft systems: Research on methods and best practices," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 233, no. 16, pp. 6044–6062, 2019.

[21] B. A. White, H.-S. Shin, and A. Tsourdos, "UAV obstacle avoidance using differential geometry concepts," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 6325–6330, 2011.

[22] H. L. N. N. Thanh and S. K. Hong, "Completion of collision avoidance control algorithm for multicopters based on geometrical constraints," *IEEE Access*, vol. 6, pp. 27111–27126, 2018.

[23] Z. Dong, Z. Chen, R. Zhou, and R. Zhang, "A hybrid approach of virtual force and A* search algorithm for uav path re-planning," in *2011 6th IEEE Conference on Industrial Electronics and Applications*, pp. 1140–1145, IEEE, 2011.

[24] S. Roelofsen, A. Martinoli, and D. Gillet, "3D collision avoidance algorithm for unmanned aerial vehicles with limited field of view constraints," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 2555–2560, IEEE, 2016.

[25] C. Zhao, J. Gu, J. Hu, Y. Lyu, and D. Wang, "Research on cooperative sense and avoid approaches based on ADS-B for unmanned aerial vehicle," in *2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC)*, pp. 1541–1546, IEEE, 2016.

[26] Y. Lin and S. Saripalli, "Sampling-based path planning for UAV collision avoidance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 11, pp. 3179–3192, 2017.

[27] Y. Chen, J. Han, and H. Wu, "Quadratic programming-based approach for autonomous vehicle path planning in space," *Chinese Journal of Mechanical Engineering*, vol. 25, no. 4, pp. 665–673, 2012.

[28] M. Radmanesh, M. Kumar, A. Nemati, and M. Sarim, "Dynamic optimal UAV trajectory planning in the national airspace system via mixed integer linear programming," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 230, no. 9, pp. 1668–1682, 2016.

[29] M. Poienariu, M. Georgescu, V. Chiroiu, and L. Munteanu, "On the dynamics of non-holonomic systems," in *Proceedings of Annual Symposium of Institute of Solid Mechanics & ACOUSTICS*, pp. 28–29, 2009.

[30] S. M. LaValle and R. Sharma, "On motion planning in changing, partially predictable environments," *The International Journal of Robotics Research*, vol. 16, no. 6, pp. 775–805, 1997.

[31] S. Petti and T. Fraichard, "Partial motion planning framework for reactive planning within dynamic environments," in *Proc. of the IFAC/AAAI Int. Conf. on Informatics in Control, Automation and Robotics*, 2005.

[32] J. J. M. Lunenburg, S. A. M. Coenen, G. J. Naus, M. J. G. van de Molengraft, and M. Steinbuch, "Motion planning for mobile robots: A method for the selection of a combination of motion-planning algorithms," *IEEE Robotics & Automation Magazine*, vol. 23, no. 4, pp. 107–117, 2016.

[33] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[34] T. Lozano-Pérez, "Spatial planning: A configuration space approach," *IEEE Trans. Computers*, vol. 32, 1983.

[35] A. Stentz *et al.*, "The focussed D* algorithm for real-time replanning," in *IJCAI*, vol. 95, pp. 1652–1659, 1995.

[36] P. Leven and S. Hutchinson, "A framework for real-time path planning in changing environments," *The International Journal of Robotics Research*, vol. 21, no. 12, pp. 999–1030, 2002.

[37] J.-C. Latombe, *Robot Motion Planning: Edition en anglais*. Springer Science & Business Media, 1991.

[38] J. T. Schwartz and M. Sharir, *General Techniques for Computing Topological Properties of Real Algebraic Manifolds*. Ablex Publishing Corporation, 1983.

[39] K. Fujimura and H. Samet, "Planning a time-minimal motion among moving obstacles," *Algorithmica*, vol. 10, no. 1, pp. 41–63, 1993.

[40] J. Van Den Berg and M. Overmars, "Planning time-minimal safe paths amidst unpredictably moving obstacles," *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1274–1294, 2008.

[41] S. M. LaValle and J. J. Kuffner Jr, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics, New Directions: The Fourth Workshop on the Algorithmic Foundations of Robotics*, pp. 293–308, 2001.

[42] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning." Citeseer, 1998.

[43] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[44] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *IEEE access*, vol. 2, pp. 56–77, 2014.

[45] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[46] R. Kindel, D. Hsu, J.-C. Latombe, and S. Rock, "Kinodynamic motion planning amidst moving obstacles," in *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, vol. 1, pp. 537–543, IEEE, 2000.

[47] S. Petti and T. Fraichard, "Safe motion planning in dynamic environments," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2210–2215, IEEE, 2005.

[48] H. Bruns, *Das Eikonal*, vol. 35. S. Hirzel, 1895.

[49] R. Kimmel, N. Kiryati, and A. M. Bruckstein, "Multivalued distance maps for motion planning on surfaces with moving obstacles," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 3, pp. 427–436, 1998.

[50] M. S. Hassouna and A. A. Farag, "Multistencils fast marching methods: A highly accurate solution to the eikonal equation on cartesian domains," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 9, pp. 1563–1574, 2007.

[51] J.-M. Mirebeau and J. Portegies, "Hamiltonian fast marching: A numerical solver for anisotropic and non-holonomic eikonal PDEs," *Image Processing On Line*, vol. 9, pp. 47–93, 2019.

[52] F. Mothes, "Trajectory planning in time-varying adverse weather for fixed-wing aircraft using robust model predictive control," *Aerospace*, vol. 6, no. 6, p. 68, 2019.

[53] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[54] H. Rohnert, "Shortest paths in the plane with convex polygonal obstacles," *Information Processing Letters*, vol. 23, no. 2, pp. 71–76, 1986.

[55] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[56] D. E. Knuth, "Big omicron and big omega and big theta," *ACM Sigact News*, vol. 8, no. 2, pp. 18–24, 1976.

[57] S. K. Ghosh, *Visibility algorithms in the plane*. Cambridge university press, 2007.

[58] D.-T. Lee and F. P. Preparata, "Euclidean shortest paths in the presence of rectilinear barriers," *Networks*, vol. 14, no. 3, pp. 393–410, 1984.

[59] J. S. Mitchell, "Shortest paths among obstacles in the plane," *International Journal of Computational Geometry & Applications*, vol. 6, no. 03, pp. 309–332, 1996.

[60] J. Hershberger and S. Suri, "An optimal algorithm for euclidean shortest paths in the plane," *SIAM Journal on Computing*, vol. 28, no. 6, pp. 2215–2256, 1999.

[61] C. Camporesi and M. Kallmann, "Computing shortest path maps with GPU shaders," in *Proceedings of the Seventh International Conference on Motion in Games*, pp. 97–102, ACM, 2014.

[62] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996.

[63] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm.," in *ICAPS*, vol. 5, pp. 262–271, 2005.

[64] International Civil Aviation Organization ICAO, "Aircraft Operations," 2006. Volume II, Construction of Visual and Instrument Flight Procedures.

[65] M. Bittner, F. Fisch, and F. Holzapfel, "A multi-model gauss pseudospectral optimization method for aircraft trajectories," in *AIAA Atmospheric Flight Mechanics Conference*, p. 4728, 2012.

[66] M. Bittner, *Utilization of Problem and Dynamic Characteristics for Solving Large Scale Optimal Control Problems*. PhD thesis, Technische Universität München, 2017.

[67] D. Marconnet, C. Norden, and L. Vidal, "Optimum use of weather radar," *Safety first*, vol. 22, pp. 22–43, 2016.

[68] Federal Aviation Administration, "Circular Advisory: Clear Air Turbulence Avoidance," 2016.

[69] M. Erdmann and T. Lozano-Perez, "On multiple moving objects," *Algorithmica*, vol. 2, no. 1-4, p. 477, 1987.

[70] M. Sauer, T. Hauf, and C. Forster, "Uncertainty analysis of thunderstorm nowcasts for utilization in aircraft routing," *4th SESAR Innovation Days (SID2014)*, pp. 1–8, 2014.

[71] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *Journal of guidance, control, and dynamics*, vol. 25, no. 1, pp. 116–129, 2002.

[72] P. Falcone, F. Borrelli, H. E. Tseng, J. Asgari, and D. Hrovat, "A hierarchical model predictive control framework for autonomous ground vehicles," in *2008 American Control Conference*, pp. 3719–3724, IEEE, 2008.

[73] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[74] K. Kober and A. Tafferner, "Tracking and nowcasting of convective cells using remote sensing data from radar and satellite," *Meteorologische Zeitschrift*, vol. 18, no. 1, pp. 75–84, 2009.

[75] A. Mirza, C. Pagé, and S. Geindre, "Flysafe–an approach to safety–using GML/XML objects to define hazardous volumes of aviation space," in *13th Conference on Aviation, Range, and Aerospace Meteorology*, 2008.

[76] Federal Aviation Administration, "Circular Advisory: Thunderstorms," 2013.

[77] M. Baldauf, J. Förstner, S. Klink, T. Reinhardt, C. Schraff, A. Seifert, K. Stephan, and D. Wetterdienst, *Kurze Beschreibung des Lokal-Modells Kürzestfrist COSMO-DE (LMK) und seiner Datenbanken auf dem Datenserver des DWD*. Deutscher Wetterdienst, 2014.

[78] M. Köhler, F. Funk, T. Gerz, F. Mothes, and E. Stenzel, "Comprehensive weather situation map based on xml-format as decision support for uavs," *Journal of Unmanned System Technology*, vol. 5, no. 1, pp. 13–23, 2017.

[79] J. Slingo and T. Palmer, "Uncertainty in weather and climate prediction," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 369, no. 1956, pp. 4751–4767, 2011.

[80] Federal Aviation Administration, Safety Team, "Thunderstorms - Don't Flirt ...Skirt'Em," 2008.

[81] "Warm weather operations: Flight hazards associated with convective weather and tropical weather systems." https://elearning.flightsafety.com/warm-weather-operations.html. Accessed: 2020-01-19.

[82] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, pp. 226–231, 1996.

[83] F. Mothes and A. Knoll, "Automatische Wettervermeidung für ein unbemanntes, solarbetriebenes Flugzeug," in *Deutscher Luft- und Raumfahrtkongress*, 2017.

[84] S. Osher and J. A. Sethian, "Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations," *Journal of computational physics*, vol. 79, no. 1, pp. 12–49, 1988.

[85] K. Alton, *Dijkstra-like ordered upwind methods for solving static Hamilton-Jacobi equations*. PhD thesis, University of British Columbia, 2010.

[86] J. Elston and E. W. Frew, "Unmanned aircraft guidance for penetration of pretornadic storms," *Journal of guidance, control, and dynamics*, vol. 33, no. 1, pp. 99–107, 2010.

[87] J. L. Bentley and T. A. Ottmann, "Algorithms for reporting and counting geometric intersections," *IEEE Transactions on computers*, no. 9, pp. 643–647, 1979.

[88] J. Hao, J. Sun, Y. Chen, Q. Cai, and L. Tan, "Optimal reliable point-in-polygon test and differential coding boolean operations on polygons," *Symmetry*, vol. 10, no. 10, p. 477, 2018.

[89] C.-W. Huang and T.-Y. Shih, "On the complexity of point-in-polygon algorithms," *Computers & Geosciences*, vol. 23, no. 1, pp. 109–118, 1997.

[90] T. Fraichard and H. Asama, "Inevitable collision states—a step towards safer robots?," *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[91] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: the international journal for geographic information and geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[92] S.-T. Wu, A. C. da Silva, and M. R. Márquez, "The douglas-peucker algorithm: sufficiency conditions for non-self-intersections," *Journal of the Brazilian Computer Society*, vol. 9, no. 3, pp. 67–84, 2004.

[93] J. E. Hershberger and J. Snoeyink, *Speeding up the Douglas-Peucker line-simplification algorithm*. University of British Columbia, Department of Computer Science, 1992.

[94] D. K. Prasad, M. K. Leung, C. Quek, and S.-Y. Cho, "A novel framework for making dominant point detection methods non-parametric," *Image and Vision Computing*, vol. 30, no. 11, pp. 843–859, 2012.

[95] M. De Berg, M. Van Kreveld, M. Overmars, and O. Schwarzkopf, "Computational geometry," in *Computational geometry*, pp. 1–17, Springer, 1997.

[96] S. Kapoor, S. N. Maheshwari, and J. S. Mitchell, "An efficient algorithm for euclidean shortest paths among polygonal obstacles in the plane," *Discrete & Computational Geometry*, vol. 18, no. 4, pp. 377–383, 1997.

[97] T. Asano, T. Asano, L. Guibas, J. Hershberger, and H. Imai, "Visibility of disjoint polygons," *Algorithmica*, vol. 1, no. 1-4, pp. 49–63, 1986.

[98] M. H. Overmars and E. Welzl, "New methods for computing visibility graphs," in *Proceedings of the fourth annual symposium on Computational geometry*, pp. 164–171, ACM, 1988.

[99] J. Kitzinger and B. Moret, *The visibility graph among polygonal obstacles: a comparison of algorithms*. PhD thesis, University of New Mexico, 2003.

[100] P. K. Agarwal, "Partitioning arrangements of lines ii: Applications," *Discrete & Computational Geometry*, vol. 5, no. 6, pp. 533–573, 1990.

[101] C. Burnikel, K. Mehlhorn, and S. Schirra, "On degeneracy in geometric computations.," in *SODA*, pp. 16–23, 1994.

[102] S. Edelkamp and S. Schroedl, *Heuristic search: theory and applications*. Elsevier, 2011.

[103] D. C. F. Rayner, M. H. Bowling, and N. R. Sturtevant, "Euclidean heuristic optimization," in *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[104] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *Journal of the ACM (JACM)*, vol. 34, no. 3, pp. 596–615, 1987.

[105] D. D. Harabor and A. Grastien, "Online graph pruning for pathfinding on grid maps," in *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[106] J. Pearl, *Heuristics: Intelligent search strategies for computer problem solving*. Addison-Wesley Pub. Co., Inc., Reading, MA, 1984.

[107] M. Helmert, G. Röger, *et al.*, "How good is almost perfect?.," in *AAAI*, vol. 8, pp. 944–949, 2008.

[108] R. E. Korf, "Recent progress in the design and analysis of admissible heuristic functions," in *International Symposium on Abstraction, Reformulation, and Approximation*, pp. 45–55, Springer, 2000.

[109] M. L. Hetland, *Python Algorithms: mastering basic algorithms in the Python Language*. Apress, 2014.

[110] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[111] H. Chitsaz and S. M. LaValle, "Time-optimal paths for a dubins airplane," in *2007 46th IEEE Conference on Decision and Control*, pp. 2379–2384, IEEE, 2007.

[112] Y. Lin and S. Saripalli, "Path planning using 3D dubins curve for unmanned aerial vehicles," in *2014 international conference on unmanned aircraft systems (ICUAS)*, pp. 296–304, IEEE, 2014.

[113] S. Eriksson-Bique, D. Kirkpatrick, and V. Polishchuk, "Discrete dubins paths," *arXiv preprint arXiv:1211.2365*, 2012.

[114] J. Brandse, M. Mulder, and M. Van Paassen, "Advanced trajectory design for the tunnel-in-the-sky display: The use of clothoids," in *AIAA guidance, navigation, and control conference and exhibit*, p. 5237, 2004.

[115] D. M. Gierszewski, V. Schneider, P. J. Lauffs, L. Peter, and F. Holzapfel, "Clothoid-augmented online trajectory generation for radius to fix turns," *IFAC-PapersOnLine*, vol. 51, no. 9, pp. 174–179, 2018.

[116] "XFLR5, analysis tool for airfoils, wings and planes operating at low reynolds numbers." http://www.xflr5.tech/xflr5.htm. Accessed: 2020-01-21.

[117] "About XFLR5 calculations and experimental measurements." http://www.xflr5.tech/docs/ Results_vs_Prediction.pdf. Accessed: 2020-02-18.

[118] E. L. Duke, R. F. Antoniewicz, and K. D. Krambeer, "Derivation and definition of a linear aircraft model," 1988.

[119] R. C. Nelson *et al.*, *Flight stability and automatic control*, vol. 2. WCB/McGraw Hill New York, 1998.

[120] A. Griewank and A. Walther, *Evaluating derivatives: principles and techniques of algorithmic differentiation*, vol. 105. Siam, 2008.

[121] A. Weber, *Methoden zur Effizienzsteigerung abstraktionsbasierter Reglerentwurfsverfahren*. Verlag Dr. Hut, 2018.

[122] J. R. Dormand and P. J. Prince, "A family of embedded runge-kutta formulae," *Journal of computational and applied mathematics*, vol. 6, no. 1, pp. 19–26, 1980.

[123] NATO Standardization Agency, "STANAG 4586 (edition 2)," 2007. ANNEX B.

[124] W. L. Dunn and J. K. Shultis, *Exploring Monte Carlo methods*. Elsevier, 2011.

[125] L. Papula, *Mathematische Formelsammlung*, vol. 4. Springer, 2017.

[126] FAA, "AC-23.1309-1E, system safety analysis and assessment for part 23 airplanes," *Federal Aviation Administration (FAA), US Department of Transportation*, vol. 17, 2011.

[127] I. Pohl, "Heuristic search viewed as path finding in a graph," *Artificial intelligence*, vol. 1, no. 3-4, pp. 193–204, 1970.

[128] M. Rieck, M. Bittner, B. Grüter, and J. Diepolder, "FALCON.m user guide. Institute of Flight System Dynamics," 2016.