aerospace

MDPI

Article

Trajectory Planning in Time-Varying Adverse Weather for Fixed-Wing Aircraft Using Robust Model Predictive Control

Federico Mothes

Department of Mechanical, Automotive and Aeronautical Engineering, University of Applied Sciences Munich, 80335 Munich, Germany; federico.mothes@hm.edu; Tel.: +49-89-1265-3352

Received: 9 April 2019; Accepted: 30 May 2019; Published: 5 June 2019

Abstract: The avoidance of adverse weather is an inevitable safety-relevant task in aviation. Automated avoidance can help to improve safety and reduce costs in manned and unmanned aviation. For this purpose, a straightforward trajectory planner for a single-source-single-target problem amidst moving obstacles is presented. The functional principle is explained and tested in several scenarios with time-varying polygonal obstacles based on thunderstorm nowcast. It is furthermore applicable to all kinds of nonholonomic planning problems amidst nonlinear moving obstacles, whose motion cannot be described analytically. The presented resolution-complete combinatorial planner uses deterministic state sampling to continuously provide globally near-time-optimal trajectories for the expected case. Inherent uncertainty in the prediction of dynamic environments is implicitly taken into account by a closed feedback loop of a model predictive controller and explicitly by bounded margins. Obstacles are anticipatory avoided while flying inside a 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.

Keywords: trajectory planning; weather avoidance; moving obstacle; model predictive control; nonholonomic constraint; fixed-wing aircraft

1. Introduction

Weather avoidance 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]. Especially thunderstorms and their surroundings are dangerous, as turbulence, gusts, wind shear, lightning, hail and icing may occur. According to the Federal Aviation Administration (FAA), the philosophy of avoidance is an integral part of flight planning [3]. This is especially true for light unmanned fixed-wing aircraft, for example high altitude pseudo-satellites [4]. Their relatively low weight and performance increase the vulnerability to adverse weather [5].

Thus far, the operation of these aircraft requires a considerable amount of manual effort in mission planning and execution. Many 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 great amount of information all at once, i.e., 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 [6], which is clearly insufficient to make optimal use of the available information. In complex scenarios with moving obstacles, this shortcoming can be 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 is fast in relation to the obstacles. However, convective weather has a high rate of change and quasi-static avoidance is prone to cause 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 [7]. The intrinsic complexity of anticipatory trajectory planning in uncertain dynamic environments calls for an automated solution. The consideration of kinematic and dynamic constraints further raises the complexity of this task [8,9].

The configuration q of an aircraft is a combination of its degrees of freedom. The configuration space C with $q \in C$ can be used for path planning in static environments. When dealing with moving obstacles, reactive planners use the static C-space to perform replanning whenever the environment changes [10,11]. However, for anticipatory trajectory planning in the presence of moving obstacles, time has to be added to the C-space. Therefore, even two-dimensional motion planning problems are difficult to solve as the dimensions are raised from two to three [12]. 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. Erdmann and Lozano-Perez [13] introduced the concept of configuration$\times$time-space, which is also called state space (X-space). Time-dependent configurations are called states $\mathrm{x} = (q, t)$ with $\mathrm{x} \in X$ [14]. Provided that a knowledge or estimate of the future obstacles exists, a sequence of states leading from start x_s to goal state x_g has to be determined. For this task, different motion planning methods exist.

An example for a control theory approach, regarding motion planning amid time-varying thunderstorms, is found in [15]. A finite horizon reach-avoid problem formulation [16] is used to maximize the probability of reaching a given point, while avoiding stochastic obstacles, under the consideration of uncertainties. Due to the curse of dimensionality [17], this kind of approach is unsuitable for fast computation of problems with high degrees of freedom.

Classic robotic approaches for motion planning amid moving obstacles include sampling-based and combinatorial motion planning. Rapidly-exploring random trees (RRTs) belong to the family of sampling-based algorithms. They 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 [18,19]. Examples for kinodynamic planning amidst moving obstacles can be found in [20–22]. However, narrow passages can pose a problem, as the likelihood for a sample being in free space decreases. Furthermore, trajectories tend to be jerky and therefore require postprocessing, e.g., by B-splines, Bézier curves or clothoids [23]. As random samples are used, the algorithm is probabilistically complete. This means that, if a trajectory exists and the number of samples approaches infinity, the probability to find the trajectory converges to one.

In [8], a combinatorial method for kinodynamic planning is presented. The upper bound for the complexity of exact motion planning in a three-dimensional dynamic environment is given by a general algorithm with a time complexity that is double exponential in the dimension of search space [24]. Canny [25] introduced an algorithm where time complexity is only exponential in its dimension. A combinatorial motion planner searches a discrete topology of free space, e.g., a visibility graph, Voronoi diagram or an exact or approximate cell decomposition. 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 parameters necessary to specify q [14]. Nonetheless, combinatorial planning offers desirable properties such as optimality, completeness and repeatable results that can be important for certification.

In this article, a combinatorial motion planner, called MPTP (see Section 2.1), is described that computes collision-free, near-time-optimal and time-monotone trajectories from x_s to x_g similar to [26,27]. Global optimization is performed using an A*-search algorithm [28]. Motion planning in the real-world is generally subject to uncertainty in environment and state predictability [14]. In the

presented setup, the planner solely relies on external environment prediction, in this case thunderstorm nowcasts. The immanent uncertainty in the environmental prediction is taken into account explicitly by bounded margins and implicitly by the closed loop of a model predictive controller (MPC). While planning takes place, information outdates in dynamic environments. This imposes a real-time decision constraint and limits the allotted time for trajectory planning [29]. For fast convergence, the X-space is iteratively built and its growth is restricted by a suitable heuristic. Furthermore, velocity and angle of climb are assumed to be at least piecewise constant. The planner uses a low-fidelity representation of aircraft dynamics with motion primitives. This ensures that the computed trajectories are compliant with the aircraft's flight envelope and resulting tracking errors are small [30].

The planner can be used as initial guess generator for gradient-based trajectory optimization in dynamic environments similar to Dubins path does in static environments to find an optimal control sequence in the homotopy class of the initial guess with a high-fidelity aircraft model [14,31].

The rest of this article is organized as follows. In Section 2, the functional principle and technical details of the presented trajectory planner for dynamic environments are described. Section 3 presents simulation results of the key capabilities, followed by the discussion (Section 4) and conclusions (Section 5).

2. Methodology

2.1. Model Predictive Trajectory Planning

The focus of this article lies on the introduction of an anticipatory trajectory planning algorithm for automated aircraft guidance through time-varying adverse weather, for example thunderstorms. Planning is performed by a reactive guidance loop, which resembles a MPC setup, due to the feedback of state and weather information. It is hence termed model predictive trajectory planner (MPTP) and provides resolution-complete and globally near-optimal trajectories for the expected case. Periodically, a new trajectory is computed, which attempts to always avoid adverse weather based on the latest nowcast. A fixed-wing aircraft has a nonholonomic turning-flight constraint and flies with finite velocity. The kinematic constraints are to avoid moving polygonal obstacles while staying within a defined mission area. In aviation, safety has the highest priority. Therefore, an optimal trajectory is defined here as the one that takes the shortest time to get to the goal while staying clear of potentially dangerous areas. As it is difficult to determine the appropriate altitude for vertical avoidance of thunderstorms (visible top is not necessarily equal to the radar top) and turbulence is frequently encountered above storm clouds, the focus lies on lateral avoidance [32].

Provided an appropriate flight controller (FC) is equipped and considering the guidance constraints in MPTP (see Section 2.3.4), it can be expected that an aircraft is able to fly the commanded trajectories. In this article, the assumption is made that commanded states are reached in due time, so that a next initial state (see Section 2.2.2) is part of the previously computed trajectory. Hence, the FC 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 Sections 2.2 and 2.3), demonstrate continuous avoidance trajectories on the basis of real thunderstorm forecasts, and compare different computation methods (see Section 3). However, in [4], the feasibility of the proposed guidance strategy is demonstrated with the setup depicted in Figure 1. The simulations are done using a six degree of freedom aircraft model, flight controller and historical wind data.

The inputs for the MPTP are the nowcast and a mission planner which specifies the reference, i.e., the goal state. The planning and control horizon of the MPTP are equal to the nowcast horizon T_N of one hour. The control output is a sequence of states, which is sent to the FC with a sampling time t_s that is equal to the nowcast update interval of $\Delta T_N = 300$ s.

The MPTP, which plans with a low-fidelity aircraft model at a low update rate, and FC, which operates on the actual aircraft at a high update rate, are decoupled as in [33,34]. The FC

is the reactive component which compensates disturbances and uncertainties and controls the aircraft in order to reach the commanded states in due time.

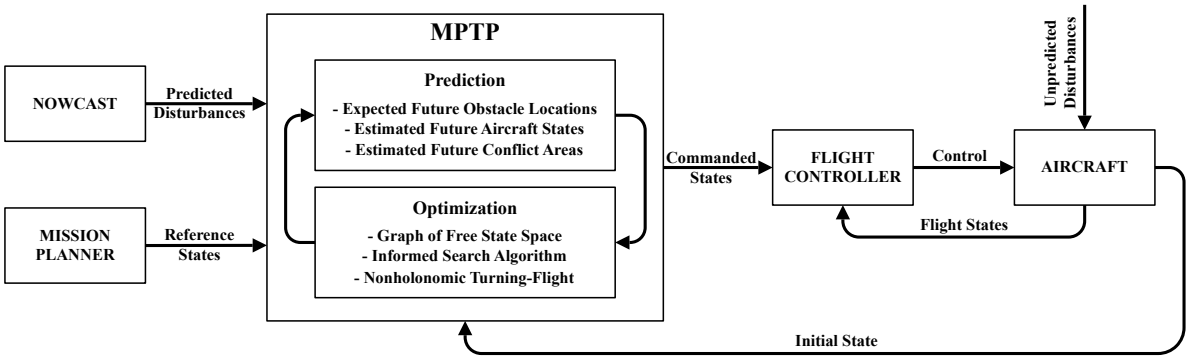


Figure 1. Schematic of the presented MPTP and its environment. The update rate is five minutes.

Thunderstorms have a short life span and their development is very difficult to predict. The uncertainty in the nowcast is implicitly considered by updates (predicted disturbances in Figure 1) and control errors by the closed loop of the MPTP (initial state in Figure 1). The recurrent correction of errors due to unpredicted disturbances yields robustness [33]. Periodical replanning of anticipatory trajectories from the actual state to the goal is done in each iteration based on the latest nowcast. Predicted thunderstorm cells are represented as spatial and temporal discrete polygonal obstacles with nonlinear motion.

The MPTP consists of a prediction and optimization module. To find a feasible trajectory to the goal, the MPTP iterates between those two (Figure 1): The prediction interprets nowcast data and estimates future conflicts between aircraft and thunderstorms (see Section 2.2) to generate a representation of free state space [14]. For the optimization, a combinatorial motion planning method with bounded uncertainty is applied (see Section 2.3). The discrete time increment Δt for prediction and optimization has to be a factor of the nowcast update interval with $\Delta T_N / \Delta t \in \mathbb{N}^+$.

2.2. Prediction–Conflict Estimation

The prediction module of the aforementioned MPTP consists of three models. The first model predicts the obstacles, i.e., the expected future thunderstorms. For this purpose, the nowcast is interpreted regarding the relative hazard for a specific aircraft taking into account the immanent uncertainty. The second prediction model computes future aircraft state samples. A third model superimposes the first two models to estimate future conflicts. As the term conflict seems more appropriate than collision in the context of weather avoidance, it is used exclusively throughout the rest of the article.

2.2.1. Expected Future Obstacle Locations

Thunderstorms and their surroundings are extremely dangerous for all kind of aircraft. The expectation of their future location is based on historical thunderstorm nowcast issued by the algorithm Radar Tracking and Monitoring (Rad-TRAM) [35]. Historical data are provided in Extensible Markup Language format (XML) by the German Aerospace Center DLR. A XML-file contains nowcasts as well as corresponding measurements. Thunderstorms are represented as polygons, which ensures small file size and allows fast processing [36]. The nowcast has a temporal horizon T_N of one hour and is updated every five minutes. Each nowcast consists of 13 sets containing the coordinates of storm cells and additional information, e.g., center of gravity, moving direction and moving speed. The first set contains the measured thunderstorm situation, while the others are predictions with an interval ΔT_N of 5 min. Information about the wind field on standard flight levels is based on COSMO-DE model [37] data and supplied in General Regularly-distributed Information in Binary form (GRIB). The forecast horizon is 21 h and the update interval is one hour [38].

In aviation a minimum lateral clearance to thunderstorms (especially cumulonimbus clouds) is recommended [32,39]. To compute safe trajectories, the weather data have to be interpreted with respect to the limitations of an aircraft. For this purpose, storm cells are expanded by appropriate margins. Two kinds of margin are applied to nowcasted cells, which are described in the following. Their purpose is to account for different kinds of uncertainty regarding the weather information.

Due to initial condition uncertainties and model errors, forecasts are never accurate [40]. Especially convective weather has a high rate of change and is difficult to predict. The forecast error generally increases with advancing time. This fact can be explicitly modeled applying bounded uncertainty, as proposed by Sauer et al. [41]. Thunderstorm cells are enlarged by nonuniform probabilistic margins. Their size is a function of a selected probability P that a cell will be included by its corresponding margin at a given time in the nowcast horizon T_N. This is a safe procedure especially when assuming high values 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 [14]. The ability to automatically plan holding patterns (Section 2.3.5) 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.

In contrast to the aforementioned margins, the safety margins account for the uncertainty regarding the presence of thunderstorm accompanying phenomena, e.g., strong winds, wind shear, turbulence (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 requires minimum distances to thunderstorms [39,42]. 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 and moving speed of a cell as these parameters allow conclusions on the potential danger. In most cases, it is safer to avoid thunderstorms on the upwind side as turbulence and hail is frequently encountered downwind [32]. By enlarging cells in these areas, the planner is tempted to shift the trajectory. By applying the aforementioned margins to concave thunderstorm polygons, they are transformed into convex polygons. This is reasonable as concave radar signatures are associated with strong turbulence and hail [32]. The superposition of probabilistic and safety margins can be seen in the Figure in Section 3.1 as light red areas around the predicted dark red thunderstorms.

High thunderstorm density and close neighboring cells indicate potentially weather active areas. They 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 [39]. An intuitive interpretation is necessary to identify weather active areas, as planning trajectories through them can cause substantial short-term changes with respect to the initial route. Discrete margins sometimes leave narrow passages between thunderstorms of which the MPTP makes use of when searching for a trajectory. To avoid this, Density Based Spatial Clustering for Applications with Noise (DBSCAN) is applied with the radius for neighborhood determination ($\epsilon \in \mathbb{R}$) set to a selected minimum corridor width [43]. In many cases, clustering of thunderstorms result in concave polygons. These are not converted to convex polygons, as otherwise excessive coverage of free space is the consequence [44].

Finally, a processed nowcast results in thirteen discrete sets of buffered thunderstorms ($t_0 + 0$ min, $t_0 + 5$ min, ..., $t_0 + 60$ min). However, the MPTP requires discrete information concerning the obstacles for the intermediate query times spaced by Δt. This is achieved by applying a linear spatiotemporal interpolation. For instance, if $\Delta t = 60$ s, four interpolated polygonal shapes for each known shape (thunderstorm) are computed for $t + 60$ s, $t + 120$ s, $t + 180$ s and $t + 240$ s. An example for this procedure can be found in [44].

2.2.2. 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 $W = 0$ or $W \neq 0$. In the real world, wind speed is rarely zero. What matters is if the aircraft or more specifically the FC can compensate the wind.

If wind can be compensated, future state primitives for up to one hour of flight can be preprocessed for different ground speeds V_G with $V_G = V_T$ for $W = 0$. For this purpose, a 3DOF simulation model of the aircraft is used. The formulas of the aircraft's motion are

$$\dot{x} = V \cos\chi \cos\gamma$$

$$\dot{y} = V \sin\chi \cos\gamma$$

$$\dot{z} = -V \sin\gamma$$

with μ being the bank angle. The derivatives of the velocity V, course or track angle χ and the angle of climb γ are given, respectively, by

$$\dot{V} = g(n_x - \sin\gamma)$$

$$\dot{\chi} = g\frac{n_z \sin\mu}{V \cos\gamma}$$

$$\dot{\gamma} = g\frac{n_z \cos\mu - \cos\gamma}{V}$$

Additionally, the horizontal and vertical load factors $n_x = \frac{T-D}{mg}$ and $n_z = \frac{L}{mg}$ are needed with T being the thrust, m the mass, g the Earth's gravitation, D the drag and L the lift.

A controller keeps the aircraft model on commanded courses between $\chi = [0° : \Delta\chi : 360°]$. The controller consists of three laws for altitude, speed and azimuth control. The connection between contemporary state samples results in isochronous curves (black lines in Figure 2a). The heart shape in Figure 2a results from turning-flight when changing the initial course $\chi_{in} = 0°$ from north by $\pm 180°$.

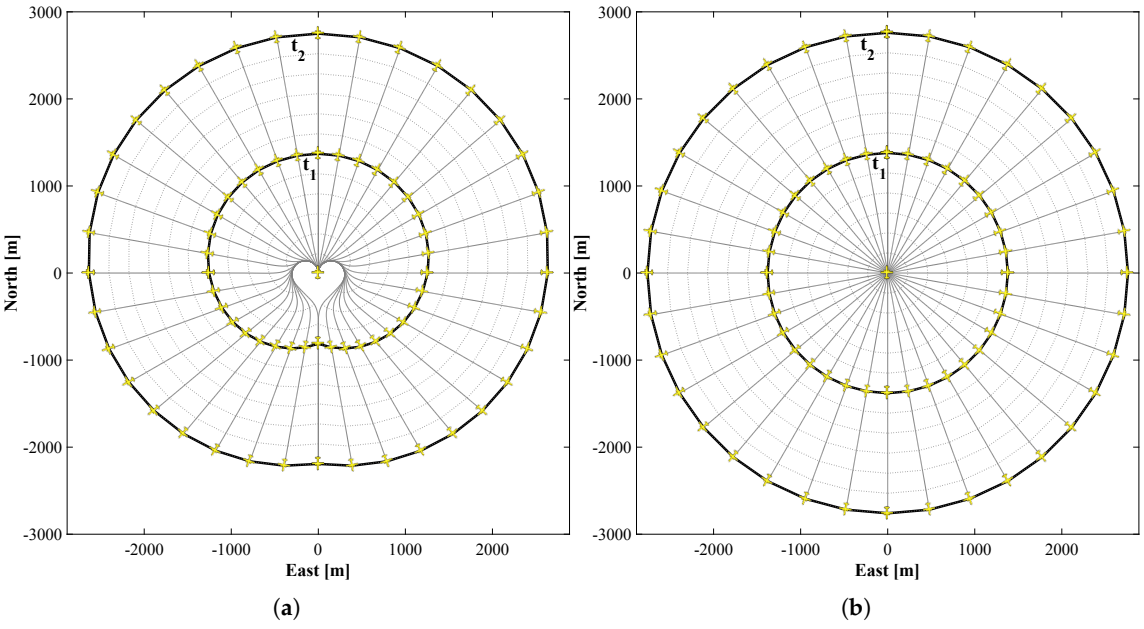


Figure 2. Future states are depicted by yellow aircraft symbols. The isochronous progress with $V_T = V_G = 23$ m/s is shown with black lines for $t_1 = 60$ s and $t_2 = 120$ s. (**a**) The results of the flight simulation with a maximum bank angle of $|\mu_{max}| = 19°$ compared to (**b**) approximated states.

The estimated set of future aircraft state samples A_{smp} are computed in a Cartesian coordinate system heading north. 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 with $A_{fut}(\mathrm{x}_{in}) = H(\mathrm{x}_{in}) \cdot A_{smp}$, written-out:

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

If the wind field is considered, future states cannot be preprocessed as the resulting V_G depends on the position and course of the aircraft. To compute a trajectory, future states have to be computed several times (see Section 2.3). The considerable computation time when using a flight simulation makes this method unsuitable for online application.

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 the initial position p_{in} (see Figure 2b). The easiest way is to set up linearly spaced matrices for each spatial dimension. The first dimension of each matrix indicates the time t and the second dimension the course χ. Alternatively, the first-order nonlinear partial differential Eikonal equation in the form $|\nabla T|F = 1$ can be applied with T being the arrival times of the expanding wave and F being a scalar speed function. If wind is not considered, 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 [45,46]. This finite-difference method is suitable for online application. Using the multistencils second order fast marching method (MSFMM) improves the accuracy considerably [47]. However, for slow aircraft, wind has a significant impact on trajectory planning, as it cannot always be completely compensated. Under the influence of wind, constant V_T results in a spatially inhomogeneous V_G profile. If the wind is considered for planning, a noniterative ordered upwind method (OUM) can be applied for a fast approximate estimation of future aircraft states [48,49].

2.2.3. Estimated Future Conflict Areas

Obstacle regions with invalid x_{fut} are called X_{obs} [14]. Originating from an initial state $\mathrm{x}_{in}(x_{in}, y_{in}, z_{in}, \chi_{in}, t_{in}) \in X$, 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 obstacle region EX_{obs}. This is done by superposition of the aforementioned prediction models for expected future obstacle location and estimated future aircraft states. The set of estimated conflicts from an initial state x_{in} is defined as the union of the set A_{fut} of future aircraft states $\mathrm{x}_{fut}(x_{fut}, y_{fut}, z_{fut}, \chi_{fut}, t_{fut})$ and set O of time-varying obstacles with arbitrary shape $O_n(t), n \in \{1, \ldots, m\}$ that depends on time t. Future states on and inside of time-varying obstacles $A_{fut}(\mathrm{x}_{in}) \cap O_n(t)$ are enclosed by concave polygons that are called estimated conflict areas (ECA) with $ECA_i(\mathrm{x}_{in}), i \in \{1, \ldots, k\}$. The estimated obstacle region is the union set $EX_{obs} = \cup_{i=1}^{k} ECA_i$. The mission area, in which the aircraft is allowed to fly, is called the workspace $W \in \mathbb{R}^2$. The free estimated state-space $EX_{free}(\mathrm{x}_{in})$ is the difference $W \setminus EX_{obs}(\mathrm{x}_{in})$.

Figure 3 shows an example for the evolution of EX_{obs} under the assumption of constant V_G. 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 2.2.1) is considered the isochronous progress is not circular anymore such as in Figure 4. The shape of the obstacle region EX_{obs} is clearly different than in Figure 3.

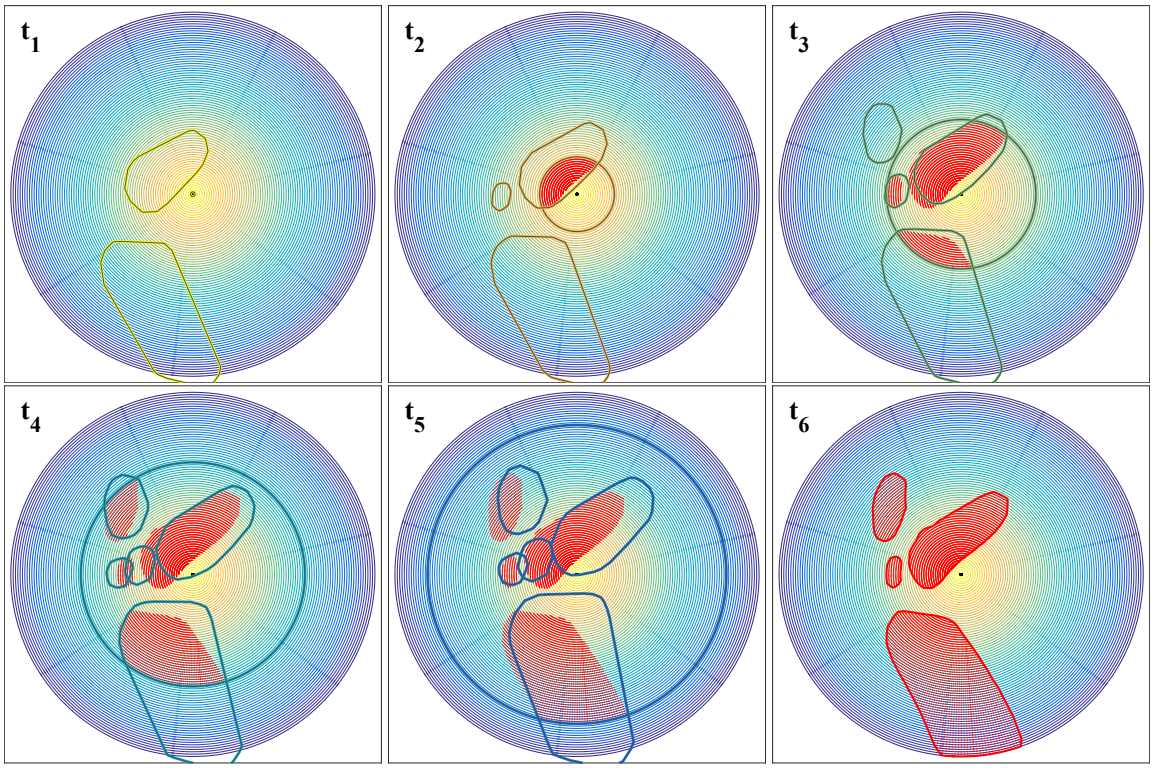


Figure 3. In this example the ground speed is equal to true airspeed. The evolution of estimated conflict areas is illustrated at six points in time (t_1–t_6). For each time, the estimated future states (highlighted isochrone circle) in conflict with contemporaneous expected thunderstorms (varying polygon shapes) are marked in red. This results in the final shape of the ECAs shown at t_6.

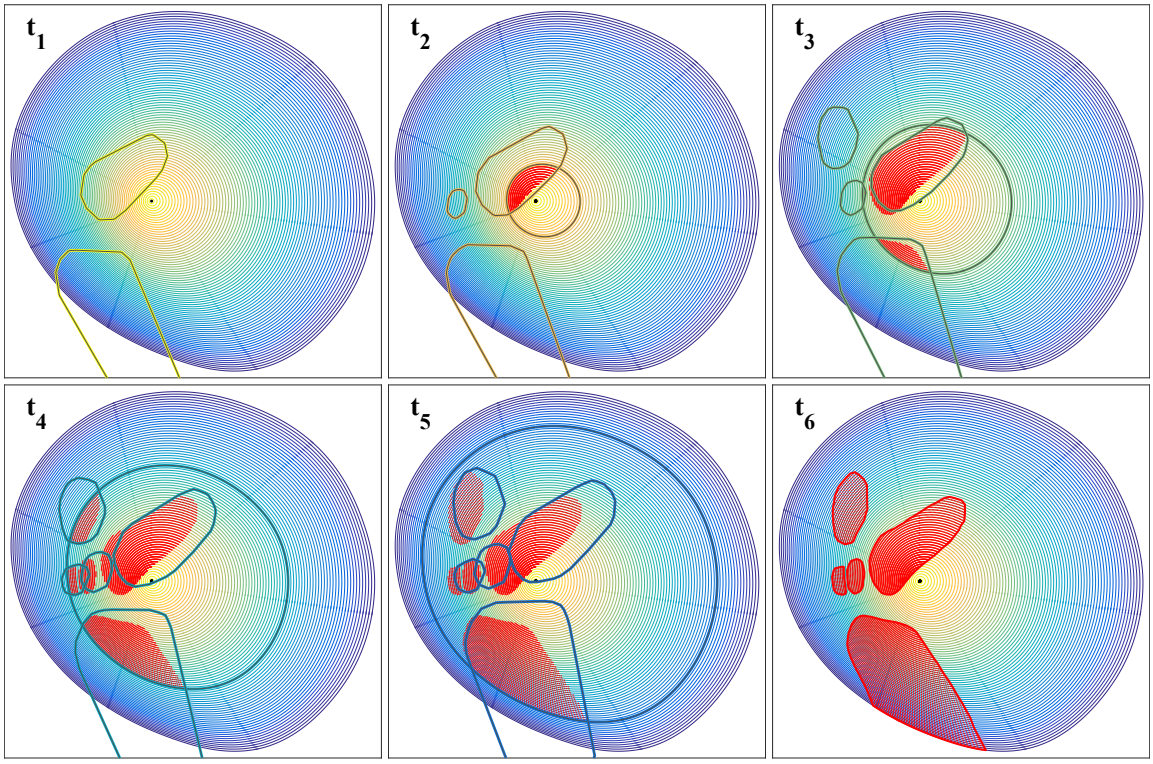


Figure 4. In the anisotropic case, the groundspeed is not equal to true airspeed and depends on the aircraft's position and course. The inhomogeneous gridded wind data are taken from a GRIB file (see Section 2.2.1). When considering wind the resulting ECAs differ noticeably from those in Figure 3.

The presented method can also be used to determine EX_{obs} in non-level flight (Figure 5). 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 non-level flight, using the methods in Section 2.3.

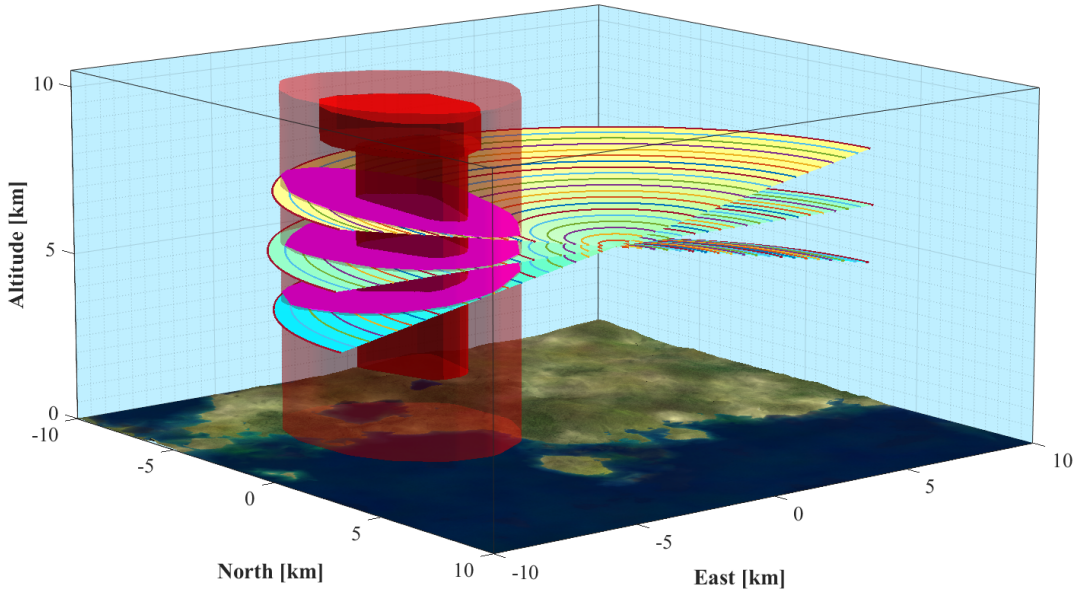


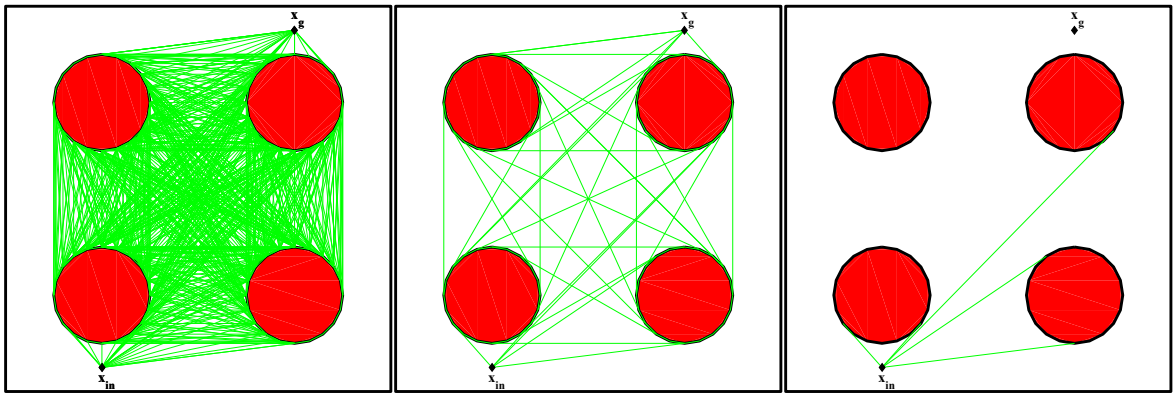
Figure 5. The ECSs (magenta surfaces) for a 3D thunderstorm cloud and its margin (translucent red volume) are shown for climb, level flight and descent.

2.3. Optimization–Trajectory Computation

This section explains the necessary steps to find a time-monotone trajectory through predicted thunderstorms under the consideration of the nonholonomic turning-flight constraint.

2.3.1. Graph of Free Estimated State Space

A visibility graph $VG(\mathrm{x}_{in}) = \{V, E\}$ is used to create a roadmap of $EX_{free}(\mathrm{x}_{in})$ (Section 2.2.3) from an initial state x_{in}. It contains vertices $v \in V$ and edges $e \in E$ which are undirected and weighted. Rohnert [50] introduced a method to compute a reduced VG for convex polygons containing exclusively tangent edges between a point and a polygon and bitangent edges between polygons. LaValle [14] described a robust method to determine bitangent edges without trigonometric functions by using cross products. After excluding all edges that are not tangent or bitangent, the remaining edges are checked for intersections. The vectorized Matlab code works for convex and nonconvex ECAs and optionally generates a partial or complete VG. A partial visibility graph $VG_p(\mathrm{x}_{in})$ only connects x_{in} to vertices $v \in V$ that are adjacent by a tangential edge. A complete visibility graph $VG(\mathrm{x}_{in})$ contains all visible edges in $EX_{free}(\mathrm{x}_{in})$. Figure 6 shows a comparison between three different VGs, inside a mission area (black square) and around obstacles (four red icosagons). The large difference in visible edges (green lines) is the reason only Method b and c are considered for trajectory planning.



(**a**) (**b**) (**c**)

Figure 6. (**a**) Unreduced; (**b**) reduced complete; and (**c**) reduced partial VG. The initial state x_{in} and goal state x_g are depicted by black diamonds.

The time complexity of the VG can be expressed by the number of vertices $|V|$. A reduction of $|V|$ in EX_{free} therefore improves the runtime of the VG. The time complexity of a graph-search algorithm (see next section) is often described by the number of vertices $|V|$ and edges $|E|$ of the searched graph, e.g., $O(|E| + |V| \log |V|)$ for Dijkstra's algorithm. A reduction in $|V|$ also reduces $|E|$ and thus speeds up the search. To reduce $|V|$ in EX_{free}, the Douglas–Peucker algorithm can be applied [51]. Through an appropriate selection of the tolerance band ϵ, a reduction can be done under preservation of shape, e.g., by an error bound termination condition [52]. Predicted thunderstorms are enlarged by probabilistic margins to model the considerable amount of uncertainty in the nowcast (see Section 2.2.1). Hence, EX_{free} is generated upon a guess of the future and does not represent an absolutely exact geometry. Therefore, small geometric inaccuracies due to line smoothing are acceptable.

2.3.2. Informed Search in a Dynamic Environment

To find anticipatory time-optimal trajectories, A*-search is applied [28]. It is suitable for single-source-single-target problems and has many advantageous properties, namely being a complete, optimal and easy to implement algorithm.

Depending on the applied heuristic (see next section), a partial or complete $VG(x_{in})$ of $EX_{free}(x_{in})$ is generated. This process is repeated until x_{in} is equal to the goal state x_g or no solution exists. In the following, the nodes of the VG are referred to as states. In contrast to the Dijkstra's algorithm, A* does not just compute the G-cost, which in this case is the time to get from the start state x_s to a candidate state x_c. Instead, it uses an additional heuristic to estimate the H-cost, which is the time to get from a x_c to x_g. The total estimated F-cost is the time it takes to get from x_s to x_g via x_c and is calculated by $F(x_c) = G(x_c) + H(x_c)$.

A complete roadmap to the goal in X-space does not exist a priori. Instead, it is iteratively generated by exploring x_{c_i} with $i \in \{1, \ldots, k\}$. The A*-search grows trajectories, which are all rooted in the start state x_s. During every loop of A*, the F-cost for each x_c is computed. The most promising x_c, which has the minimum estimated F-cost, is explored in the next iteration. The bookkeeping is done in the Open List (OL). For fast computation, the OL is a priority queue implemented as binary heap that contains all x_{c_i} sorted in ascending F-cost order as well as their parent states x_{p_i}.

At the beginning of the algorithm, the only state in the OL is x_s, which automatically has the lowest estimated F-cost. The $EX_{free}(x_{in})$ is computed from $x_{in} = x_s$. Next, a VG generates a roadmap $VG(x_{in})$ of the static $EX_{free}(x_{in})$. Subsequently, the adjacent states of x_s are inserted into OL as x_c. Then, x_s is popped from the priority queue and inserted into the so-called Closed List (CL), which stores already explored states. This process repeats until the OL is empty, in which case the algorithm reports that no trajectory to the goal exists and performs partial planning, or the goal is inserted into the CL,

in which case the trajectory is constructed by backtracing the corresponding parent states and is then commanded to the FC.

The following pseudocode (Algorithm 1) illustrates the functionality of the MPTP including the A*-search.

Algorithm 1: MPTP

Input: initial state $x_{in}(x_{in}, y_{in}, z_{in}, \chi_{in}, t_{in})$ and latest thunderstorm nowcast (see Figure 1)
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, merge or cluster the polygons
  perform spatiotemporal interpolation for the query times spaced by Δt (see Section 2.2.1)
  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**
    /* start of the A*-like-search*/
    initalize the Open List $OL = \{x_{in}\}$ with the start state x_s set as initial state x_{in}
    initalize empty Closed List $CL = \emptyset$
    while $OL \neq \emptyset$ **do**
      compute the set of estimated future states $A_{fut}(x_{in})$ for constant V_T
      compute the obstacle region $EX_{obs}(x_{in}) = A_{fut}(x_{in}) \cap O_n(t)$
      compute the free state estimated space $EX_{free}(x_{in}) = W \setminus EX_{obs}(x_{in})$
      generate the auxiliary state(s) x_{aux} (see upcoming Section 2.3.4)
      generate roadmap of free state space $\{V, E\} = VG(EX_{free}(x_{in}), x_{in}, x_g, x_{aux})$
      select candidate states x_c adjacent to x_{in} that meet nonholonomic constraint (see upcoming Section 2.3.4)
      foreach x_c **do**
        calculate total cost to get from x_s to x_g via x_c with $F(x_c) = G(x_c) + H(x_c)$
      end
      the new x_{in} is the minimum F-cost state from the OL
      remove x_{in} from the OL and insert it into the CL
      if x_{in} *is the goal state* x_g **then**
        break
      end
    end
    if $OL = \emptyset$ **then**
      perform partial planning, e.g., continue on last trajectory, avoid inevitable conflict areas (comparable to [22])
    else
      construct the anticipatory trajectory by backtracing of the parent states x_p
      command trajectory states to the flight controller (FC) (see Figure 1)
    end
  else
    command holding pattern in free state space to the FC (see upcoming Section 2.3.5)
  end
end

In the pseudocode above, the typical relaxation of the A*-search is missing (inside the while loop). In a static environment, A*-algorithm checks if new candidate nodes are already in the OL or CL. If their G-cost is lower, the existing entry is updated by replacing the G-cost and parent node.

However, in the presented case, the obstacles move. Each x_c is tangential to the ECAs whose form is uniquely valid from the perspective of their $x_p = x_{in}$. Therefore, even if a x_c has the same q as an explored state and a lower time t from x_s to x_c (which is the G-cost), it cannot replace the state in the OL or CL. As each x_c is adjacent to its x_p by a straight line. the final trajectory is guaranteed to be optimal.

2.3.3. Heuristics for the Informed Search

In this section, three different heuristics are described. The quality of the heuristic is essential for the convergence of the A*-search and the optimality of the computed trajectories. For trajectory planning, the H-cost is the estimated time it takes to get from a candidate x_c to the goal state x_g. If this cost is underestimated, the solution is optimal but unnecessary search is done. A heuristic is admissible if it never overestimates costs as this can lead to suboptimal results.

The first and most ineffective heuristic is to set H to a constant value, e.g., equal to zero. Then, A*-search basically becomes Dijkstra's algorithm and the search expands as a wavefront in all directions.

The Euclidean distance heuristic (EDH) estimates the H-costs, to get from each x_c to x_g, by the beeline distance. It is an admissible and effective heuristic [14]. When Dijkstra or A* with EDH is applied, only a partial VG_p with the first order adjacent states has to be computed. This saves computation time in comparison to the generation of a complete VG. However, EDH ignores obstacles as it only evaluates the beeline to the goal.

Finally, a new shortest static path heuristic (SSPH) is introduced in this article. A complete $VG(x_{in})$ of every $EX_{free}(x_{in})$ is searched by a nested A*-search. This inner A* for his part applies the EDH to determine the H-cost for every new x_c by computing its shortest path to x_g (which can be converted to time as V_T is constant) in the static $EX_{free}(x_{in})$. Alternatively, the Fast Marching Method can be applied (see Section 2.2.2). In this case, no VG is needed, which is especially interesting when searching three-dimensional space. Although SSPH is computationally more intensive than EDH, the explicit consideration of obstacles speeds up the search (see Section 3). However, SSPH can overestimate the H-cost as it is computed in the $VG(x_p)$, which is why optimality of trajectories cannot be guaranteed. Hence, SSPH is a non-admissible heuristic. In practice, Dijkstra, EDH and SSPH produce identical or similar trajectories (see Section 3).

2.3.4. Modeling Nonholonomic Turning-Flight

A fixed-wing aircraft has velocity constraints in the y- and z-direction of its body-fixed frame of reference and cannot fly directly sideways or upwards. While the number of velocities of the aircraft is reduced, there are no restrictions in its configurations as they can be reached performing a series of maneuvers. Therefore, a fixed-wing aircraft is a nonholonomic system [53].

If the future state primitives from Section 2.2.2 are used, the nonholonomic turning-flight constraint is considered innately. For the methods that approximate future states, the nonholonomic constraint has to be explicitly modeled. As the combinatorial planning approach does not model the nonholonomic turning-flight by default, an auxiliary method is required. Additional states, which in the following are referred to as auxiliary states x_{aux}, are generated to obtain discrete curvature-constrained trajectories, which can be interpreted as a discrete version of Dubins path [54]. An x_{aux} can be generated for one turning-sense (unidirectional turning-flight) or both turning-senses (bidirectional turning-flight). A maximum course angle change $|\Delta\chi_{max}|$ from χ_{in} is introduced. Adjacent future states to x_{in} that require $|\Delta\chi| \geq |\Delta\chi_{max}|$ are disconnected from x_{in} in the VG. The value for $|\Delta\chi_{max}|$ cannot be chosen freely, as it has to be ensured that the aircraft is able to reach commanded states x_{aux} in due time.

A straightforward method for the computation of auxiliary states x_{aux} is introduced in the following. Starting from x_{in}, the initial χ_{in} can be changed by $0° \leq |\Delta\chi_{max}| \leq 90°$. An x_{aux} is then 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. 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 greater or equal than the minimum stabilization distance MSD [55]. 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 the 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 V_T is $\leq V_{cor}$. The V_{cor} has to be greater than the stall speed V_S and less or equal to maximum velocity V_{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 7, a normalized minimum stabilization distance MSD_{norm} and velocity correction δV are determined as a function of $|\Delta\chi_{max}|$ (see Tables 1 and 2), by setting the minimum turning radius to $R_{min} = 1$ (unit turning-circle). To determine the smallest possible $MSD = MSD_{norm} R_{min}$ for fly-over states, for both the roll-in and roll-out radius the same radius $R_{min} = 1$ is applied (see Figure 7b). Generally, roll-in and roll-out radius are different, leading to a smooth trajectory but longer MSD.

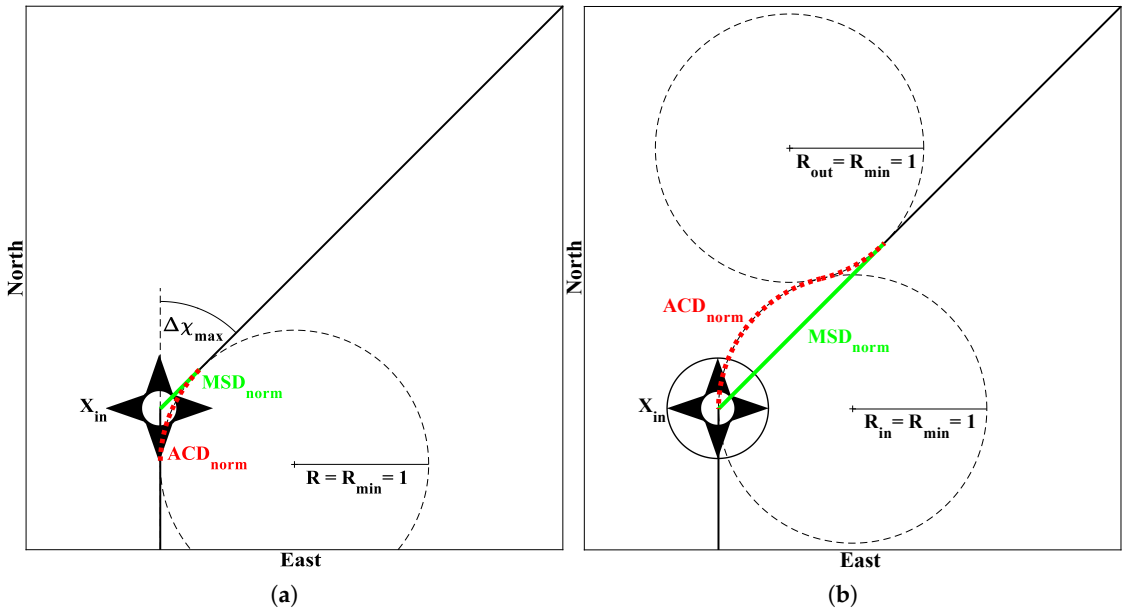


Figure 7. Normalized aircraft covered distance and normalized minimum stabilization distance, when x_{in} is performed (**a**) as fly-by or (**b**) as fly-over state.

The values for the MSD_{norm} (Tables 1 and 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 s for a fly-over state [55].

As the time increment Δt is a fixed value, the relative 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 is twice the MSD_{norm} (see Figure 7) and the velocity correction factor is computed by

$$\delta V = 1 + \frac{ACD_{norm} - 2MSD_{norm}}{2MSD_{norm}}.$$

The formula for a fly-over state is

$$\delta V = 1 + \frac{ACD_{norm} - MSD_{norm}}{MSD_{norm}}.$$

Table 1. Normalized minimum stabilization distances 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 2. Normalized minimum stabilization distances 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

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

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

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

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 3 shows exemplary values for different $|\mu_{max}|$.

Table 3. Values for minimum stabilization distances for $V_T = 80$ m/s, $|\Delta\chi_{max}| = 45°$ (as used in Section 3) 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 flyable, the minimum time increment Δt is determined by

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

Notice that the values for the velocity correction δV in Tables 1 and 2 only apply for the special case, in which the leg distance LD is equal to the MSD. Else 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

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

For the example in Section 3.3, 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 2 is $\delta V = 1.078$. Applying the aforementioned formula with a 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 Section 3 satisfy the criteria for flyability, i.e., $MSD \leq LD = V_T \Delta t$ and $V_S < V_{cor} < V_{max}$.

To increase the robustness of the MPTP, the maximum course change can be defined as soft constraint. A x_{aux} is only valid if it is neither in nor on $EX_{obs}(x_{in})$. 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 $MSD \leq LD$. If possible, $|\Delta\chi_{max}|$ should be kept small, as this generally results in smoother and more optimal trajectories.

2.3.5. Automatic 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 [56]. 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, a unidirectional turning constraint is applied, for example only turn right (UPS-method). This prevents the MPTP to plan meandering trajectories, however they may no longer be near-optimal.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.

For example, by setting $|\Delta\chi_{max}| = 45°$, a circular holding pattern is described by the x_{aux}, as inscribed regular n-gon with $n = 8$ (see Figure 8 and Section 3.3). The distance on a straight segment of an n-gon is given by $S_n = V_T \Delta t$. For fly-over states, the radius R_{cc} of the corresponding circumcircle, on whose orbit the aircraft flies, is

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

The distance on the corresponding circumcircle segment is

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

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

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

The mean corrected velocity is calculated by

$$V_{cor} = V_T \delta V(n).$$

To ensure the feasibility of a planned trajectory, it is important that $V_S \leq V_{cor} \leq V_{max}$ and $R_{min}(V_{cor})$ is $\leq R_{cc}$. In the upcoming example in Section 3.3, the velocity for planning is $V_T = 80$ m/s.

Therefore, the corrected velocity is $V_{cor} = V_T \delta V(8) = 80 \cdot 1.026$ m/s $= 82.09$ m/s. The required bank angle for the coordinated turn is

$$\mu = \arctan \frac{V_{cor}^2}{gR} = 6.25°.$$

This leaves reserves regarding the turning-flight performance. In theory, the flight controller is able to put the presented guidance strategy into practice, arriving at the commanded waypoints in due time.

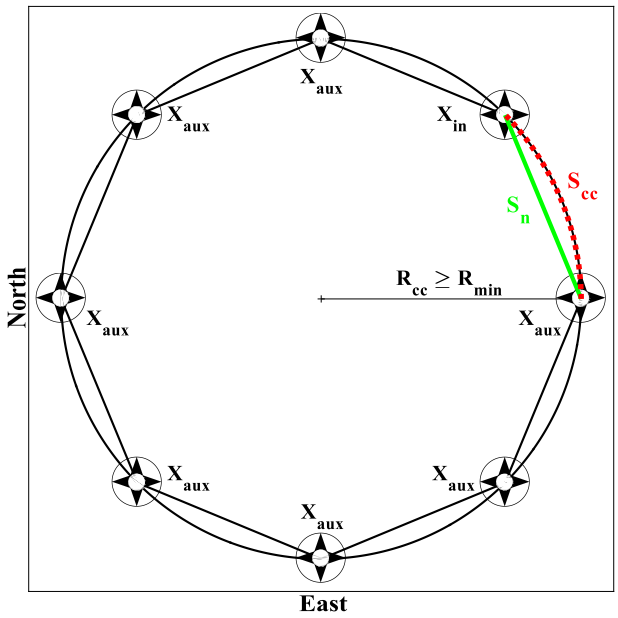


Figure 8. Circular holding pattern defined by eight fly-over states.

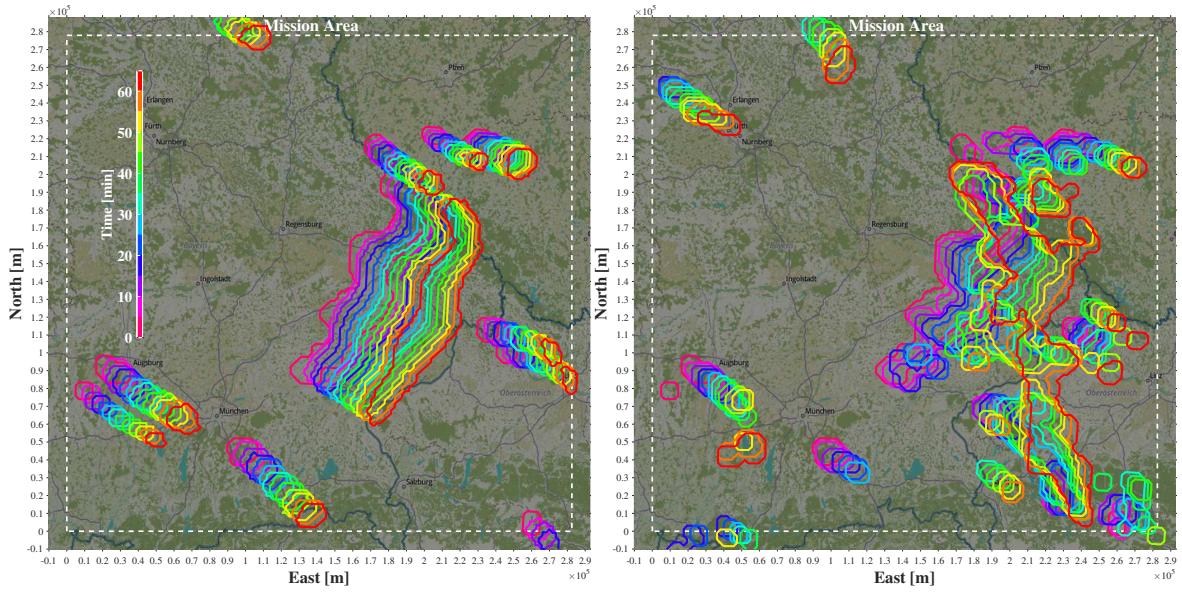
2.3.6. Planning with Moving Goal

If the position of the goal varies with time, the goal becomes 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 1). The reactive guidance is likely to result in a trajectory with pursuit curve.

If x_g can be predicted by the mission planner, 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 right Figure in Section 3.4).

3. Results

The simulation results were generated using Matlab R2015b code, Intel Core i7-6700 CPU @ 3.40 GHz and 32 GB RAM running on 64-Bit-Windows 7. As the algorithm always performed the same calculation steps and number of iterations, the computational times were averaged over 10 runs. The mission area (white dashed lines in Figures 9–15) extended from 47.50° to 50.00° in latitude and 10.25° to 14.00° in longitude. The origin of the coordinate system in Figures 9–15 was at 47.50° latitude and 10.25° longitude. It was assumed that the wind can be compensated by the FC. The maximum bank angle of the aircraft was $|\mu_{max}| = 25°$ and the maximum velocity was $V_{max} = 110$ m/s. For the following two avoidance scenarios, thirteen historical nowcasts by Rad-TRAM from 27 June 2015 were used. All figures have the same color-code for real time. The period from t_0 =19:05 h to 20:05 h UTC was selected due to the significant divergence between nowcast and measured weather. Figure 9a shows the nowcast issued at t_0 in the nowcast period of t_0 to $t_0 + 60$ min, while Figure 9b shows the measured data for the same instants but in real-time.



(**a**) (**b**)

Figure 9. (**a**) Thunderstorm prediction by the nowcast issued 19:05 h for the next hour; and (**b**) subsequent real measurements for the same period of time, both issued by Rad-TRAM.

3.1. Anticipatory Trajectory Planning

An anticipatory trajectory was computed based on the expectation of how the future will look like (Section 2.2.3). Figure 10 shows an exemplary trajectory based on the nowcast from t_0 =19:05 h (see Figure 9a). It is the first trajectory of the continuous avoidance example in Figure 12 (magenta line). The vertical dimension is the nowcast time T_N (Figure 10). 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 from Section 2.2.1. Their size varies from 10,000 m at t_0 to 35,000 m at $T_N = 3600$ s and defines the regions to be avoided. The aircraft flies with constant $V_T = V_G = 80$ m/s. Future states were sampled on a circular grid with $\Delta\chi = 2°$ using the approximate method from Section 2.2.2.

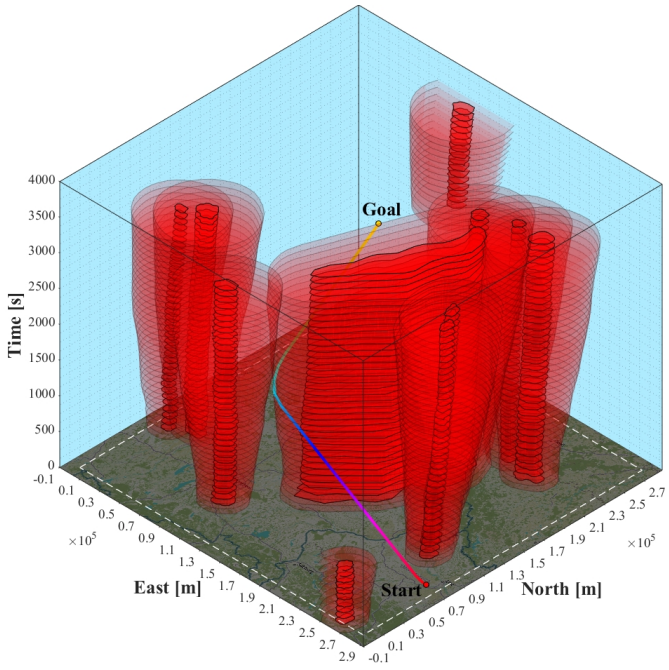


Figure 10. Near-optimal anticipatory avoidance trajectory based on 19:05 h nowcast.

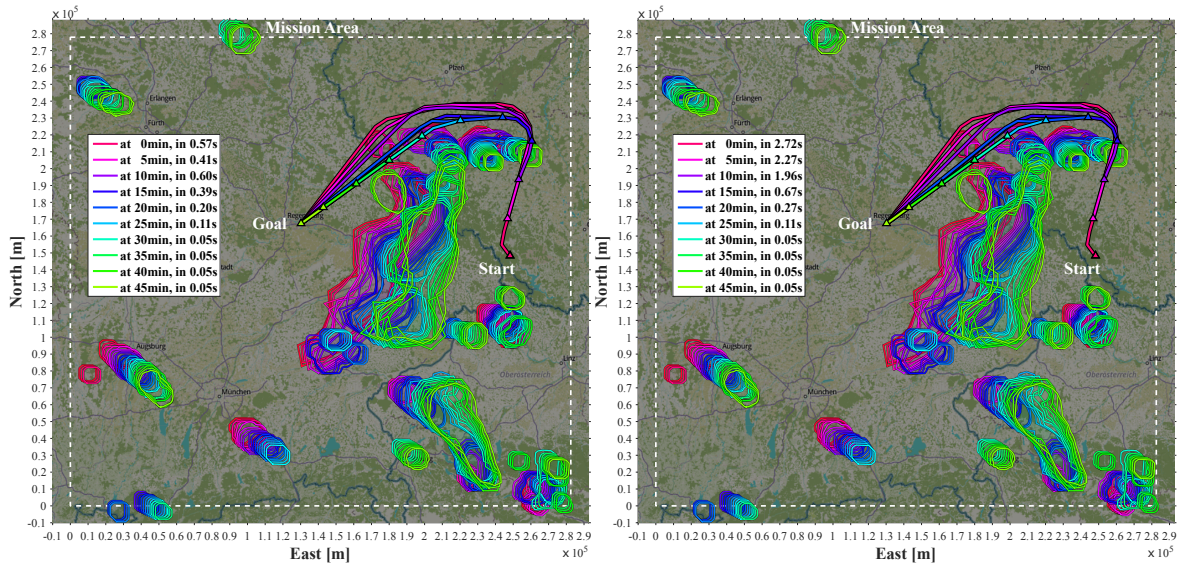
Continuous avoidance was achieved by recurrent replanning based on the latest environment prediction. Therefore, more examples of anticipatory trajectories can be found in Figures 11 and 12 (monochrome lines).

3.2. Continuous Avoidance Scenarios with Comparison of Different Heuristics

In this section, two avoidance scenarios are presented in which the aircraft has to fly towards the developing thunderstorms to get to the goal. Both are critical cases as uncertainty in the nowcast is generally most pronounced in moving direction [41]. None of the trajectories was postprocessed, e.g., by B-splines. The monochrome colored lines in Figures 11 and 12 are anticipatory trajectories that were computed by the MPTP starting from the triangles of the same color. They mark the 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.

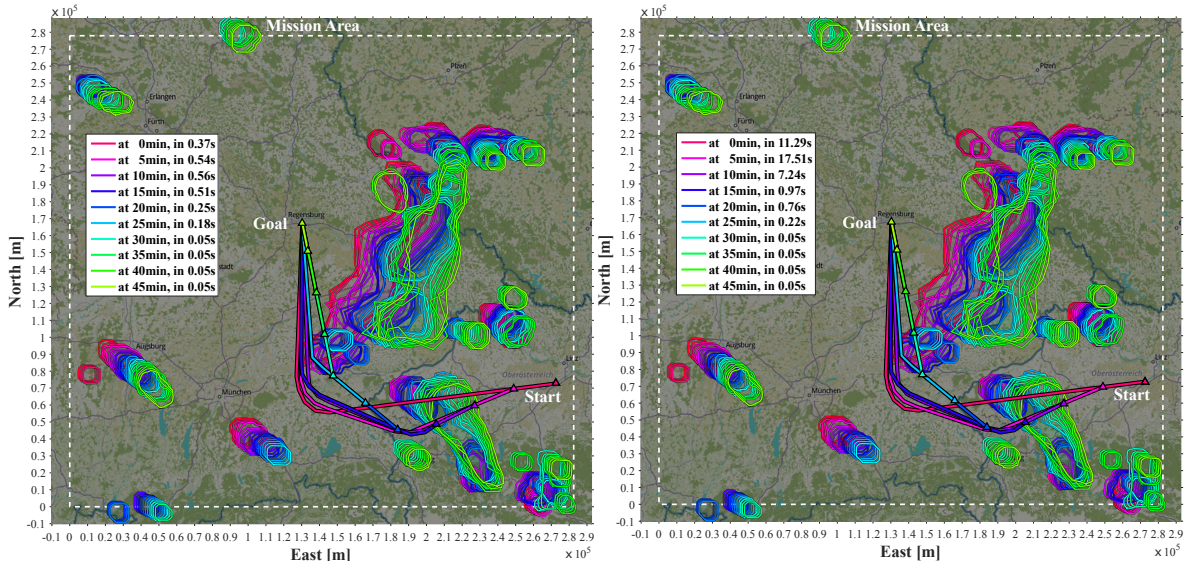
Figures 11 and 12 show anticipatory trajectories and the resulting trajectories both, respectively, generated using SSPH and EDH (Section 2.3.3) for visual comparison. The runtimes for each MPTP iteration are plotted in the legends on the top left side. Runtimes for the weather processing depend on the data and take around 0.9–2.5 s without clustering. They are not added to the runtimes for trajectory planning as the weather processing is done in advance.

Additionally, a constant H-cost of zero was applied, which degrades the A*- to the Dijkstra-algorithm. Random perturbations of aircraft states were intentionally omitted 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 was computed with unidirectional or bidirectional turning-flight (Section 2.3.4).



(**a**) (**b**)

Figure 11. (**a**) Continuous avoidance using SSPH. (**b**) Continuous avoidance using EDH. The runtimes for the MPTP iterations are listed in the respective legends.



(**a**) (**b**)

Figure 12. (**a**) Continuous avoidance using SSPH. (**b**) Continuous avoidance using EDH.

3.2.1. Scenario 1

Table 4 lists the values used for the simulation of Scenario 1.

Table 4. Index s stands for start state and index g for the goal state.

lon_s	lat_s	χ_s	lon_g	lat_g	t_0	Δt	V_T	$\lvert\Delta\chi_{max}\rvert$	ϵ
13.543°	48.834°	280°	11.980°	49.006°	19:05 h	100 s	80 m/s	45°	2000 m

The trajectory lengths are the same in all MPTP iterations using Dijkstra and A* with EDH. However, the lengths are listed in the tables below as only nine out of ten trajectories have identical lengths when using A* with SSPH. In the fourth MPTP iteration, the trajectory with SSPH is about 200 m longer than with A* with EDH or Dijkstra. The reason for this discrepancy and the following results are discussed in Section 4.

Unidirectional Turning-Flight: The results in Figure 11 and Tables 5–8 were computed with turning constrained to right-sense. This means that in the main loop of A*-search (see Section 2.3.2) for every explored state an x_{aux} to the right is added to the OL (as described in Section 2.3.4).

Table 5. Dijkstra applied to the avoidance in Figure 11, if the aircraft can only turn right.

MPTP Iteration	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Trajectory Length [km]	247.0	219.6	192.6	159.5	134.5	108.6	84.80	61.70	38.60	15.40
Trajectory States (TRST)	13	11	9	9	2	2	1	1	1	1
Explored States (EXST)	628	710	996	420	230	96	115	8	5	2
TRST/EXST Ratio [%]	2.070	1.549	0.904	2.143	0.870	2.083	0.870	12.50	20.00	50.00

Table 6. A* with **EDH** applied to the avoidance in Figure 11, if the aircraft can only turn right.

MPTP Iteration	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Trajectory Length [km]	247.0	219.6	192.6	159.5	134.5	108.6	84.80	61.70	38.60	15.40
Trajectory States (TRST)	13	11	9	9	2	2	1	1	1	1
Explored States (EXST)	58	49	50	14	5	2	1	1	1	1
TRST/EXST Ratio [%]	22.41	22.45	18.00	64.29	40.00	100.0	100.0	100.0	100.0	100.0

Table 7. A* with **SSPH** applied to the avoidance in Figure 11, if the aircraft can only turn right.

MPTP Iteration	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Trajectory Length [km]	247.0	219.6	192.6	159.7	134.5	108.6	84.80	61.70	38.60	15.40
Trajectory States (TRST)	13	11	9	6	2	2	1	1	1	1
Explored States (EXST)	13	11	14	6	3	2	1	1	1	1
TRST/EXST Ratio [%]	100.0	100.0	64.29	100.0	100.0	100.0	100.0	100.0	100.0	100.0

Table 8. Ratio between explored states for **Dijkstra** vs. **A*** with **SSPH** and **A*** with **EDH** vs. **A*** with **SSPH**, if the aircraft can only turn right.

MPTP Iteration	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
EXST(Dijkstra)/EXST(SSPH)	48.31	64.55	71.14	70.00	76.67	48.00	115.0	8.000	5.000	2.000
EXST(EDH)/EXST(SSPH)	4.462	4.455	3.571	2.333	1.667	1.000	1.000	1.000	1.000	1.000

Bidirectional Turning-Flight: The aircraft can turn left and right which ensures that the trajectory contains no unnecessary loops. Tables 9–11 contain results for the case that in the while loop of the A*-search (see Section 2.3.2) for every explored state two x_{aux} to the right and left are added to the OL (as described in Section 2.3.4). 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 9. A* using **EDH** applied to the avoidance in Figure 11, if the aircraft can turn left and right.

MPTP Iteration	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Trajectory Length [km]	247.0	219.6	192.6	159.5	134.5	108.6	84.80	61.70	38.60	15.40
Trajectory States (TRST)	13	11	9	9	2	2	1	1	1	1
Explored States (EXST)	12159	1397	113	14	5	2	1	1	1	1
TRST/EXST Ratio [%]	0.107	0.787	7.964	64.29	40.00	100.0	100.0	100.0	100.0	100.0

Table 10. A* using **SSPH** applied to the avoidance in Figure 11, if the aircraft can turn left and right.

MPTP Iteration	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Trajectory Length [km]	247.0	219.6	192.6	159.7	134.5	108.6	84.80	61.70	38.60	15.40
Trajectory States (TRST)	13	11	9	6	2	2	1	1	1	1
Explored States (EXST)	15	11	14	6	3	2	1	1	1	1
TRST/EXST Ratio [%]	86.67	100.0	64.29	100.0	66.67	100.0	100.0	100.0	100.0	100.0

Table 11. Ratio between explored states for **A*** with **EDH** vs. **A*** with **SSPH**, if the aircraft can turn left and right.

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

3.2.2. Scenario 2

Table 12 lists the values used for the simulation of Scenario 2.

Table 12. Index s stands for start state and index g for the goal state.

lon_s	lat_s	χ_s	lon_g	lat_g	t_0	Δt	V_T	$\lvert\Delta\chi_{max}\rvert$	ϵ
13.868°	48.154°	205°	11.980°	49.006°	19:05 h	100 s	80 m/s	84°	3000 m

Using identical parameters produces identical anticipatory trajectories in all MPTP iterations when applying Dijkstra, A* with EDH and A* with SSPH. Their 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 following results are discussed in Section 4.

Unidirectional Turning-Flight: The results in Figure 12 and Tables 13–16 are computed with turning constrained to right-sense. This means that in the main loop of the A*-search (see Section 2.3.2) for every explored state an x_{aux} to the right is added to the OL (as described in Section 2.3.4).

Table 13. Dijkstra applied to the avoidance in Figure 12, if the aircraft can only turn right.

MPTP Iteration	1	2	3	4	5	6	7	8	9	10
Trajectory States (TRST)	6	8	8	8	3	3	1	1	1	1
Explored States (EXST)	17,633	63,940	174,137	6960	768	145	24	19	6	3
TRST/EXST Ratio [%]	0.034	0.013	0.005	0.115	2.391	2.069	4.167	5.263	16.67	33.33

Table 14. A* with **EDH** applied to the avoidance in Figure 12, if the aircraft can only turn right.

MPTP Iteration	1	2	3	4	5	6	7	8	9	10
Trajectory States (TRST)	6	8	8	8	3	3	1	1	1	1
Explored States (EXST)	231	346	145	18	13	4	1	1	1	1
TRST/EXST Ratio [%]	2.597	2.312	5.517	44.44	23.08	75.00	100.0	100.0	100.0	100.0

Table 15. A* with **SSPH** applied to the avoidance in Figure 12, if the aircraft can only turn right.

MPTP Iteration	1	2	3	4	5	6	7	8	9	10
Trajectory States (TRST)	6	8	8	8	3	3	1	1	1	1
Explored States (EXST)	6	10	11	9	4	3	1	1	1	1
TRST/EXST Ratio [%]	100.0	80.00	72.73	88.89	75.00	100.0	100.0	100.0	100.0	100.0

Table 16. Ratio between explored states for **Dijkstra** vs. **A*** with **SSPH** and **A*** with **EDH** vs. **A*** with **SSPH**, if the aircraft can only turn right.

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

Figure 13 visualizes the success rate (TRST/EXST Ratio) or rather convergence of A*-search with the proposed SSPH in Figure 13a versus EDH in Figure 13b.

Bidirectional Turning-Flight: The aircraft can turn left and right which ensures that the trajectory contains no unnecessary loops. Tables 17–19 contain results for the case that in the main loop of the A*-search (see Section 2.3.2) for every explored state two x_{aux} to the right and left are added to the OL (as described in Section 2.3.4). Dijkstra is again not listed for comparison as the first MPTP iteration does not converge after hours.

Table 17. A* with **EDH** applied to the avoidance in Figure 12, if the aircraft can turn left and right.

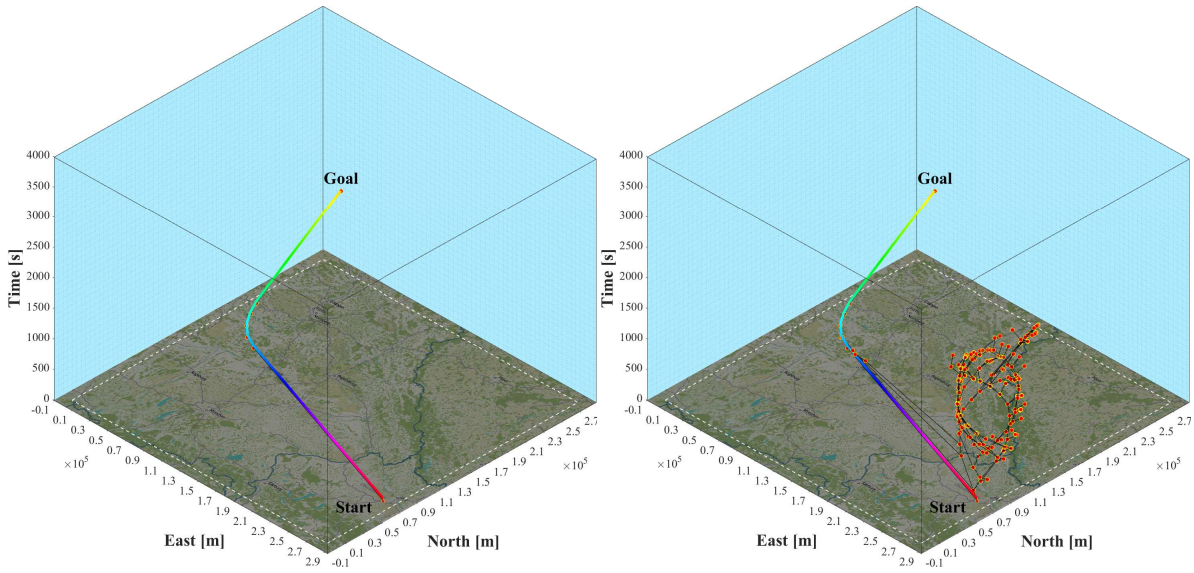
MPTP Iteration	1	2	3	4	5	6	7	8	9	10
Trajectory States (TRST)	6	8	8	8	3	3	1	1	1	1
Explored States (EXST)	28,007	13,809	1267	67	19	5	1	1	1	1
TRST/EXST Ratio [%]	0.021	0.058	0.631	11.94	15.79	60.00	100.0	100.0	100.0	100.0

Table 18. A* with **SSPH** applied to the avoidance in Figure 12, if the aircraft can turn left and right.

MPTP Iteration	1	2	3	4	5	6	7	8	9	10
Trajectory States (TRST)	6	8	8	8	3	3	1	1	1	1
Explored States (EXST)	6	10	11	9	4	3	1	1	1	1
TRST/EXST Ratio [%]	100.0	80.00	72.73	88.89	75.00	100.0	100.0	100.0	100.0	100.0

Table 19. Ratio between explored states for **A*** with **EDH** vs. **A*** with **SSPH**, if the aircraft can turn left and right.

MPTP Iteration	1	2	3	4	5	6	7	8	9	10
EXST(EDH)/EXST(SSPH)	4668	1381	115.2	7.444	4.750	1.667	1.000	1.000	1.000	1.000



(**a**) (**b**)

Figure 13. Comparison for the first MPTP iteration in Tables 14 and 15: candidate future states (yellow and red dots) in the Closed List using (**a**) **A*** with **SSPH** and (**b**) **A*** with **EDH**. Until the goal is reached, the A*-like algorithm incrementally generates ramifications in the X-space all rooted in the start state.

3.3. Automatic Holding Pattern Scenario

Table 20 lists the values used for the simulation of an automatic holding pattern.

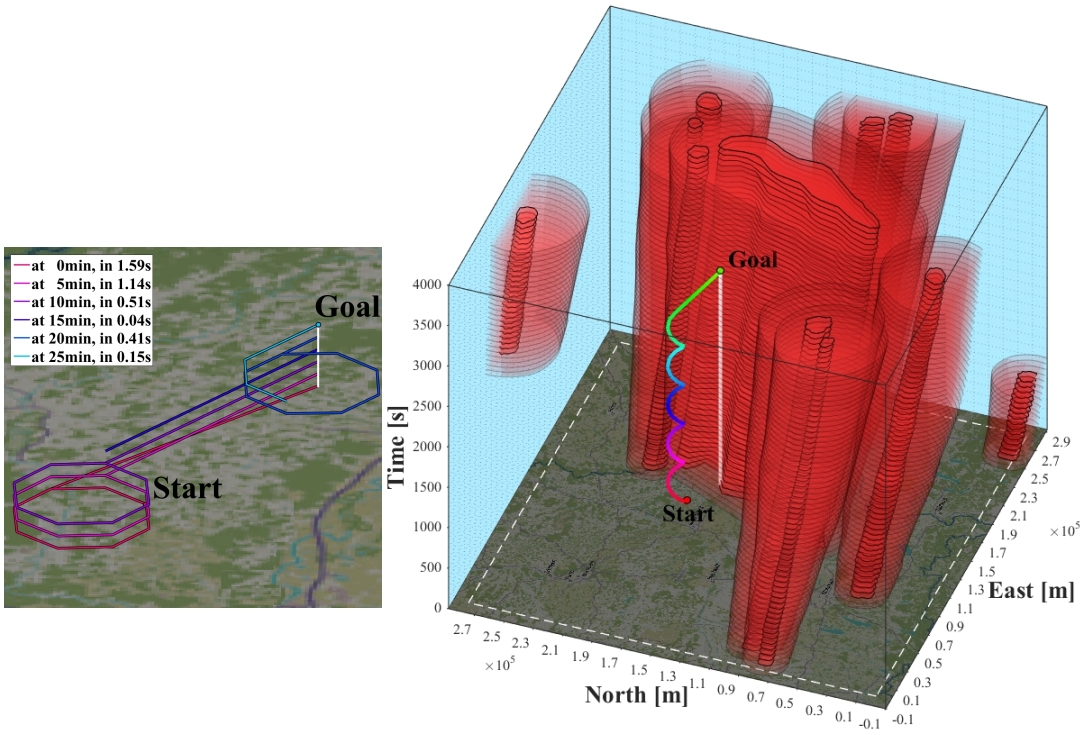
Table 20. Index s stands for start state and index g for the goal state.

lon_s	lat_s	χ_s	lon_g	lat_g	t_0	Δt	V_T	$\lvert\Delta\chi_{max}\rvert$	ϵ
12.168°	49.134°	220°	11.980°	49.006°	19:05 h	60 s	80 m/s	45°	2000 m

Figure 14a 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 are identical to 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 14b). 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 at $t_3 = t_0 + 15$ 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 at the actual time of arrival of 19:34 h.



(**a**) (**b**)

Figure 14. (**a**) Anticipatory trajectories including automatic holding patterns with runtimes for the MPTP iterations in the legend. (**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.

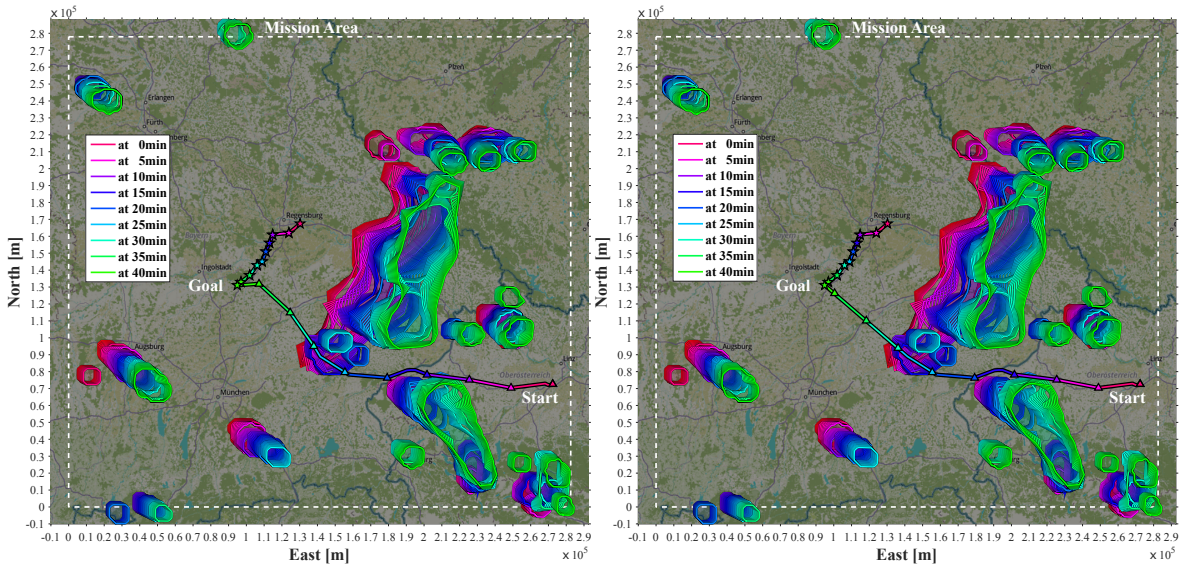
3.4. Moving Goal Scenario

Table 21 lists the values used for the simulation of an automatic holding pattern.

Table 21. Index s stands for start state and index g for the goal state.

lon_s	lat_s	χ_s	lon_g	lat_g	t_0	Δt	V_T	$\lvert\Delta\chi_{max}\rvert$	ϵ
13.868°	48.154°	205°	11.980°	49.006°	19:05 h	30 s	80 m/s	45°	2000 m

In Figure 15a, the actual position of the moving goal (colored star markers) is updated with every nowcast which results in a reactive navigation. In Figure 15b, the aircraft has the information about the future goal states in advance, which allows a tactical flight directly towards the estimated area of meeting.



(**a**) (**b**)

Figure 15. (**a**) Reactive trajectory and (**b**) anticipatory trajectory to a moving goal.

4. Discussion

Comparing Figure 9a,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 11 and 12. Although solely relying on external information by nowcast, the MPTP safely avoids thunderstorms in all presented scenarios (Sections 3.2–3.4). 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 10 illustrates that the 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 V_G the vertical slope is likewise constant.

The final trajectories from start state to goal in Scenario 1 and Scenario 2 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, the final trajectory is in both scenarios shorter than expected in the first MPTP iteration. Probabilistic margins help to prevent the MPTP from replanning substantially diverging trajectories.

In Scenario 2, the goal lies in the direction of the moving thunderstorms (Figure 12). At 19:05 h, the thunderstorms are moving at a ground speed ranging from 6.2 m/s to 17.6 m/s and a mean of 10.3 m/s with 88.5° to 128.5° and a mean of 110° direction from north. The beeline between start and goal is 170.9 km and almost in the opposite direction at 303.7°. 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 and 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 [42]. 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 availability of an onboard radar.

The results in Section 3.2 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 2.3.4) is added into the OL in each while loop of the A*-search. 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. However, it is important to keep the number of additional states in the OL as small as possible.

The performance of A*-search with three different heuristics is compared in Section 2.3.3. 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 a 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 allowed Dijkstra does not converge in hours which is why in both scenarios no data are presented. In this case, the minimum outdegree of each state is two. Therefore, the number of states to explore increases exponentially, which additionally slows down priority queue operations. 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 Scenario 1, the TRST/EXST hit ratio for unidirectional turning is $\leq$2.2% when obstacles are in between start and goal (Table 5). In Scenario 2, the maximum allowed deviation is larger than in Scenario 1, which causes more candidate states in the OL. The TRST/EXST ratio in the first four MPTP iterations is extremely poor (Table 13). In the third MPTP iteration number of EXST is 15,830 times that of A*-search with SSPH (Table 16). These results indicate that Dijkstra is unsuitable for the present application in X-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 on par with SSPH in uni- an bidirectional flight (see last MPTP iterations in Tables 8, 11, 16 and 19). Although the pruning of nontangent edges in the VG prevents unnecessary candidate states, the search with EDH can be rather slow especially in crowded environments. The reason is that the beeline distance to the goal ignores obstacles. In unidirectional turning 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 Scenario 1 (Table 8) A* with EDH explores at most 4.46 times more states than A* with SSPH and in Scenario 2 at most 38.5 times (Table 16). 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 Tables 11 and 19. The search is unwilling to increase the H-cost in order to fly around obstacles and instead 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 into account implicitly. As only the most promising states are explored, the performance between uni- and bidirectional turning flight is almost identical. Only in Scenario 1 in the first MPTP iteration the bidirectional search explores two states more than in the unidirectional case (compare Tables 7 and 10). In Scenario 2, the number of EXST is identical for all MPTP iterations (compare Tables 15 and 18). The introduced SSPH is not admissible as overestimation of the heuristic cost is possible, which is why the optimality of the trajectories cannot be guaranteed. This is due to the fact that SSPH is computed in the graph of free state space from the perspective of an 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 candidate states 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. However, overestimation of the H-cost is partly done on purpose by an inflation factor $\epsilon \geq 1$ as this can speed up the search [57,58]. In Scenario 1, in the fourth iteration of the MPTP, the computed trajectory with SSPH (Table 7) is 200 m longer

than with Dijkstra (Table 5) or A* with EDH (Table 6). 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. In Section 3, the aircraft travels at $V_T = V_G = 80$ m/s. For example, in the first MPTP iteration in Scenario 2, the aircraft covers a distance of 80 m/s·0.42 s = 29.6 m using A* with SSPH vs. 80 m/s·11.29 s= 903.2 m using A* with EDH while the trajectory is computed.

As it may not be possible to find a complete trajectory in an allotted time interval, an additional partial planning strategy is needed to avoid passivity [29] and guarantee decisioning. A possible strategy is to perform holdings in free state space. The example in Section 3.3 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 2.2.1) sometimes conservatively cover free space including the goal which reduces the probability of finding a solution. As before, the final trajectory takes less time than first guessed. Due to periodic replanning (Section 2.1) and estimation of future aircraft states (Section 2.2.2), the planner has the innate ability to deal with a moving goal (Section 3.4).

5. Conclusions

A method for robust trajectory planning in uncertain dynamic environments is 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. The algorithm always performs the same computations and therefore the results are repeatable. Due to anticipatory planning and consideration of nonholonomic turning-flight constraint of fixed-wing aircraft, computed trajectories do not require postprocessing, as demonstrated in [4]. The ability to automatically plan holding patterns, if the goal itself or the access is covered by obstacles, improves the success rate of the planner.

The performance of A* search with three different heuristics was compared. Using A*-search with EDH provides the same results as Dijkstra's algorithm but is much faster. A*-search with the introduced SSPH is faster than A* with EDH in all presented examples. This highly targeted heuristic enables fast and reliable combinatorial trajectory planning in state space. Although SSPH is not admissible and optimality cannot be guaranteed, the trajectories are near-optimal and mostly identical to those computed with Dijkstra or A*-search with EDH.

There is still room for improvement and open questions. An optimization and implementation of the algorithm in C/C++ will improve the runtime considerably. The MPTP is sensitive to certain parameters that 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 is necessary to get good results. Planned research will also include stochastic analysis regarding the reliability or success rates of the MPTP. Furthermore, an extension to four-dimensional trajectory planning using the fast marching method is intended. In addition, the ability to plan trajectories for multiple aircraft is on the agenda.

Funding: This research was funded as part of the Ludwig Bölkow Campus project StraVARIA by the Project Management Agency for Aeronautics Research and Technology of the Bavarian Ministry of Economic Affairs, Regional Development and Energy.

Acknowledgments: The author would like to thank his family and in alphabetical order of first names, Alexander Knoll, Alexander Weber, Ferdinand Settele, Marcus Kreuzer and Reiko Müller for their great support.

Conflicts of Interest: The author declares no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Jenamani, R.K.; Kumar, A. Bad weather and aircraft accidents—Global vis-à-vis Indian scenario. *Curr. Sci.* **2013**, *104*, 316–325.
2. Fultz, A.J.; Ashley, W.S. Fatal weather-related general aviation accidents in the United States. *Phys. Geogr.* **2016**, *37*, 291–312. [CrossRef]
3. Federal Aviation Administration. *Circular Advisory: Clear Air Turbulence Avoidance*; Federal Aviation Administration: Washington, DC, USA, 2016.
4. Müller, R.; Kiam, J.J.; Mothes, F. Multiphysical simulation of a semi-autonomous solar powered high altitude pseudo-satellite. In Proceedings of the 2018 IEEE Aerospace Conference, Big Sky, MT, USA, 3–10 March 2018; pp. 1–16.
5. Li, G.; Baker, S.P. Crash risk in general aviation. *JAMA* **2007**, *297*, 1596–1598. [CrossRef] [PubMed]
6. Miller, G.A. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychol. Rev.* **1956**, *63*, 81. [CrossRef] [PubMed]
7. De Lellis, E.; Morani, G.; Corraro, F.; Di Vito, V. On-line trajectory generation for autonomous unmanned vehicles in the presence of no-fly zones. *Proc. Inst. Mech. Eng. Part G J. Aerosp. Eng.* **2013**, *227*, 381–393. [CrossRef]
8. Canny, J.; Donald, B.; Reif, J.; Xavier, P. On the complexity of kinodynamic planning. In Proceedings of the 1988 29th Annual Symposium on Foundations of Computer Science, White Plains, NY, USA, 24–26 October 1988.
9. Donald, B.; Xavier, P.; Canny, J.; Reif, J. Kinodynamic motion planning. *J. ACM* **1993**, *40*, 1048–1066. [CrossRef]
10. Stentz, A. The focussed D* algorithm for real-time replanning. *IJCAI* **1995**, *95*, 1652–1659.
11. Leven, P.; Hutchinson, S. A framework for real-time path planning in changing environments. *Int. J. Robot. Res.* **2002**, *21*, 999–1030. [CrossRef]
12. Mitchell, J.S. Geometric shortest paths and network optimization. In *Handbook of Computational Geometry*; CiteSeer: New York, NY, USA, 2000, Volume 334, pp. 633–702.
13. Erdmann, M.; Lozano-Perez, T. On multiple moving objects. *Algorithmica* **1987**, *2*, 477. [CrossRef]
14. LaValle, S.M. *Planning Algorithms*; Cambridge University Press: Cambridge, UK, 2006.
15. Hentzen, D.; Kamgarpour, M.; Soler, M.; Gonzalez-Arribas, D. On maximizing safety in stochastic aircraft trajectory planning with uncertain thunderstorm development. *Aerosp. Sci. Technol.* **2018**, *79*, 543–553. [CrossRef]
16. Summers, S.; Kamgarpour, M.; Lygeros, J.; Tomlin, C. A stochastic reach-avoid problem with random obstacles. In Proceedings of the 14th international conference on Hybrid Systems: Computation and Control, Chicago, IL, USA, 12–14 April 2011; pp. 251–260.
17. Bellman, R. *Dynamic Programming*; Princeton University Press: Princeton, NJ, USA, 1957.
18. LaValle, S.M. *Rapidly-Exploring Random Trees: A New Tool for Path Planning*; Citeseer: New York, NY, USA, 1998.
19. LaValle, S.M.; Kuffner, J.J., Jr. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics, New Directions: The Fourth Workshop on the Algorithmic Foundations of Robotics*; A. K. Peters, Ltd.: Natick, MA, USA 2001; pp. 293–308.
20. LaValle, S.M.; Kuffner, J.J., Jr. Randomized kinodynamic planning. *Int. J. Robot. Res.* **2001**, *20*, 378–400. [CrossRef]
21. Kindel, R.; Hsu, D.; Latombe, J.C.; Rock, S. Kinodynamic motion planning amidst moving obstacles. In Proceedings of the IEEE International Conference on Robotics and Automation, San Francisco, CA, USA, 24–28 April 2000; Volume 1, pp. 537–543.
22. Petti, S.; Fraichard, T. Safe motion planning in dynamic environments. In Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, AB, Canada, 2–6 August 2005; pp. 2210–2215.
23. Elbanhawi, M.; Simic, M. Sampling-based robot motion planning: A review. *IEEE Access* **2014**, *2*, 56–77. [CrossRef]
24. Schwartz, J.T.; Sharir, M. *General Techniques for Computing Topological Properties of Real Algebraic Manifolds*; Ablex Publishing Corporation: New York, NY, USA, 1983.

25. Canny, J. *The Complexity of Robot Motion Planning*; MIT Press: Cambriage, MA USA, 1988.
26. Fujimura, K.; Samet, H. Planning a time-minimal motion among moving obstacles. *Algorithmica* **1993**, *10*, 41–63. [CrossRef]
27. Van Den Berg, J.; Overmars, M. Planning time-minimal safe paths amidst unpredictably moving obstacles. *Int. J. Robot. Res.* **2008**, *27*, 1274–1294. [CrossRef]
28. Hart, P.E.; Nilsson, N.J.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.* **1968**, *4*, 100–107. [CrossRef]
29. Petti, S.; Fraichard, T. Partial motion planning framework for reactive planning within dynamic environments. In Proceedings of the IFAC/AAAI International Conference on Informatics in Control, Automation and Robotics, Barcelona, Spain, 14–17 Setember 2005.
30. Bottasso, C.L.; Leonello, D.; Savini, B. Path planning for autonomous vehicles by trajectory smoothing using motion primitives. *IEEE Trans. Control Syst. Technol.* **2008**, *16*, 1152–1168. [CrossRef]
31. Bittner, M. Utilization of Problem and Dynamic Characteristics for Solving Large Scale Optimal Control Problems. Ph.D. Thesis, Technische Universität München, München, Germany, 2017.
32. Marconnet, D.; Norden, C.; Vidal, L. Optimum use of weather radar. *Saf. First* **2016**, *22*, 22–43.
33. Frazzoli, E.; Dahleh, M.A.; Feron, E. Real-time motion planning for agile autonomous vehicles. *J. Guid. Control. Dyn.* **2002**, *25*, 116–129. [CrossRef]
34. Falcone, P.; Borrelli, F.; Tseng, H.E.; Asgari, J.; Hrovat, D. A hierarchical model predictive control framework for autonomous ground vehicles. In Proceedings of the 2008 American Control Conference, Seattle, WA, USA, 11–13 June 2008; pp. 3719–3724.
35. Kober, K.; Tafferner, A. Tracking and nowcasting of convective cells using remote sensing data from radar and satellite. *Meteorol. Z.* **2009**, *18*, 75–84. [CrossRef]
36. Mirza, A.; Pagé, C.; Geindre, S. FLYSAFE—An approach to safety—Using GML/XML objects to define hazardous volumes of aviation space. In Proceedings of the 13th Conference on Aviation, Range, and Aerospace Meteorology, New Orleans, LA, USA, 20–24 January 2008.
37. Baldauf, M.; Förstner, J.; Klink, S.; Reinhardt, T.; Schraff, C.; Seifert, A.; Stephan, K.; Wetterdienst, D. *Kurze Beschreibung des Lokal-Modells Kürzestfrist COSMO-DE (LMK) und seiner Datenbanken auf dem Datenserver des DWD*; Deutscher Wetterdienst: Offenbach, Germany, 2014.
38. Köhler, M.; Funk, F.; Gerz, T.; Mothes, F.; Stenzel, E. Comprehensive weather situation map based on XML-format as decision support for UAVs. *J. Unman. Syst. Technol.* **2017**, *5*, 13–23.
39. Federal Aviation Administration. *Circular Advisory: Thunderstorms*; Federal Aviation Administration: Washington, DC, USA, 2013.
40. Slingo, J.; Palmer, T. Uncertainty in weather and climate prediction. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **2011**, *369*, 4751–4767. [CrossRef] [PubMed]
41. Sauer, M.; Hauf, T.; Forster, C. Uncertainty Analysis of Thunderstorm Nowcasts for Utilization in Aircraft Routing. In Proceedings of the 4th SESAR Innovation Days (SID2014), Madrid, Spain, 25–27 November 2014; pp. 1–8.
42. Federal Aviation Administration, Safety Team. *Thunderstorms—Don't Flirt ...Skirt'Em*; Federal Aviation Administration, Safety Team: Washington, DC, USA, 2008.
43. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, 2–4 August 1996, pp. 226–231.
44. Mothes, F.; Knoll, A. Automatische Wettervermeidung für ein unbemanntes, solarbetriebenes Flugzeug. In Proceedings of the German Aerospace Congress, Darmstadt, Germany, 30 September–2 October 2017.
45. Osher, S.; Sethian, J.A. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *J. Comput. Phys.* **1988**, *79*, 12–49. [CrossRef]
46. Sethian, J.A. A fast marching level set method for monotonically advancing fronts. *Proc. Natl. Acad. Sci. USA* **1996**, *93*, 1591–1595. [CrossRef] [PubMed]
47. Hassouna, M.S.; Farag, A.A. Multistencils fast marching methods: A highly accurate solution to the eikonal equation on cartesian domains. *IEEE Trans. Pattern Anal. Mach. Intell.* **2007**, *29*, 1563–1574. [CrossRef] [PubMed]
48. Alton, K. Dijkstra-Like Ordered Upwind Methods for Solving Static Hamilton-Jacobi Equations. Ph.D. Thesis, University of British Columbia, Vancouver, BC, Canada, 2010.

49. Elston, J.; Frew, E.W. Unmanned aircraft guidance for penetration of pretornadic storms. *J. Guid. Control. Dyn.* **2010**, *33*, 99–107. [CrossRef]
50. Rohnert, H. Shortest paths in the plane with convex polygonal obstacles. *Inf. Process. Lett.* **1986**, *23*, 71–76. [CrossRef]
51. Douglas, D.H.; Peucker, T.K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartogr. Int. J. Geogr. Inf. Geovis.* **1973**, *10*, 112–122. [CrossRef]
52. Prasad, D.K.; Leung, M.K.; Quek, C.; Cho, S.Y. A novel framework for making dominant point detection methods non-parametric. *Image Vis. Comput.* **2012**, *30*, 843–859. [CrossRef]
53. Latombe, J.C. *Robot Motion Planning: Edition en Anglais*; Springer Science & Business Media: Berlin, Germany, 1991.
54. Eriksson-Bique, S.; Kirkpatrick, D.; Polishchuk, V. Discrete dubins paths. *arXiv* **2012**, arXiv:1211.2365.
55. International Civil Aviation Organization ICAO. *Aircraft Operations. Volume II, Construction of Visual and Instrument Flight Procedures*; International Civil Aviation Organization ICAO: Montreal, QC, Canada, 2006.
56. NATO Standardization Agency. *STANAG 4586*, 2nd ed.; ANNEX B; NATO Standardization Agency: Brussels, Belgium, 2007;
57. Pohl, I. Heuristic search viewed as path finding in a graph. *Artif. Intell.* **1970**, *1*, 193–204. [CrossRef]
58. Pearl, J. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*; Addison-Wesley Pub. Co., Inc.: Reading, MA, USA, 1984.