Guidance, Navigation, and Control and Co-located Conferences
August 19-22, 2013, Boston, MA
AIAA Guidance, Navigation, and Control (GNC) Conference

Multi-query Path Planning for an Unmanned Fixed-Wing Aircraft

Moritz Niendorf*
University of Michigan, Ann Arbor, Michigan, 48109, USA
Fabian Schmitt†
University of the German Armed Forces, Neubiberg, 85579, Germany
Florian-Michael Adolf‡
DLR German Aerospace Center, Braunschweig, 38108, Germany

Path planning for fixed-wing aircraft in obstacle rich environments requires the consideration of the motion constraints of the aircraft. Most existing motion planning algorithms for systems with dynamics are single query planners, which leads to a large computation overhead if multiple path planning problems are solved. This paper presents a roadmap-based multi-query path planner which generates feasible paths for a simplified dynamics model of a fixed-wing unmanned aircraft. Due to the multi-query property multiple path queries can be solved efficiently once the roadmap is constructed. Quasi-random Hammersley sets are used to place samples in a four dimensional configuration space to account for the motion constraints of the aircraft, which is modeled by the superposition of a Dubins vehicle in the horizontal plane and a double integrator with state and input constraint in the vertical plane. The optimization of roadmap parameters and post-processing of paths are discussed in detail. Simulation results show the applicability of the suggested planner.

I. Introduction

Small unmanned fixed-wing aircraft can operate at low altitudes while preserving the beneficial range and endurance properties of fixed-wing aircraft in comparison to rotorcrafts. However, due to the nonholonomic constraints caused by the dynamics of the aircraft path planning for low flying fixed-wing aircraft in obstacle rich environment such as urban terrain requires addressing the constraints at an early stage of the path planning process. Hence, a feasible path is a path that is collision free within a given clearance distance, reaches the desired goal positions and is flyable by the aircraft, i.e. feasible with respect to the dynamics of the aircraft. Low altitude operation is desirable for purposes of for example terrain masking or applications which require a sensor to be within range of targets on the ground such as for example communication with unattended ground sensors.

DLR Institute of Flight Systems develops multiple autonomous flying platforms and performs research aiming at maximized onboard information processing, decision-making and flight control. Prometheus (see fig.1) is a fixed-wing unmanned aircraft (UA) with 25 kg maximum takeoff weight.[1] A roadmap-based multi-query-path planner is part of the onboard mission planning and execution framework (MiPlEx)[2,3] developed for the ARTIS (Autonomous Rotorcraft Testbed for Intelligent Systems) rotorcraft UA family.[4] Based upon this planner we develop and implement a multi-query path planning algorithm for the operation of a low flying fixed-wing unmanned aircraft. The multi-query property of the planning method supports solving combinatorial optimization problems.

*Graduate Student, now: Department of Aerospace Engineering, University of Michigan, AIAA Member.
†Researcher, now: Department of Aerospace Engineering, University of the German Armed Forces Munich.
‡Researcher, Institute of Flight Systems, Unmanned Aircraft Department, Braunschweig, Senior AIAA Member.

The objective of this paper is to illustrate the implementation of the roadmap path planner for the Prometheus unmanned fixed-wing aircraft. However, the approach could be applied to virtually any fixed-wing aircraft with similar motion constraints.

The remainder of this paper is organized as follows: The next section introduces the problem in more detail and derives requirements for the planning algorithm. An analysis of related work then leads to choice of the path planning method and the description of the suggested roadmap planning approach and post-processing steps. Subsequently the choice of key parameters is presented and simulations are carried out to verify the choice of parameters and show further optimization of the algorithm.

II. Problem Description

Planning paths for unmanned fixed-wing aircraft operating close to the ground in obstacle rich environment is a challenging problem which can be defined as follows:

Given the workspace denoted $\mathcal{W}$ which is $\mathbb{R}^3$, the obstacle space $\mathcal{O}$ which is a subset of $\mathcal{W}$, find a path, i.e. a sequence of configurations q in the free configuration space $\mathcal{C}_{\text{free}}$ from an initial configuration q_0 to a goal configuration q_s. A configuration q consists of the position of the vehicle in $\mathcal{W}$ and an its orientation given by the Euler angles. The set of all possible configurations form the configuration space $\mathcal{C}$ which can be divided into the free configuration space $\mathcal{C}_{\text{free}}$ and the set of all configurations that are in collision with an obstacle $\mathcal{C}_{\text{obs}}$. The aircraft is subject to nonholonomic and kinodynamic constraints, where nonholonomic constraints refer to nonintegrable velocity constraints on $\mathcal{C}$ and kinodynamic constraints are second-order constraints,[5] i.e. acceleration constraints. Such path must not contain inevitable collision states (ICS),[6] which are states, i.e. configurations, velocities and accelerations of which the configuration $q(t_0)$ is $q(t_0) \in \mathcal{C}_{\text{free}}$ but necessarily lead to collision, i.e. $q(t_0 + \delta t) \in \mathcal{C}_{\text{obs}}$ due to the acceleration and velocity at t_0.

Generally, computing shortest paths in the 3-D space with polyhedral obstacles is NP-hard.[7]

Additionally, a typical UA mission comprises not only one goal but multiple goal positions. Thus, scheduling of multiple waypoints needs typically to be accomplished. For example the exact solution of a prototypical task scheduling problem, the traveling salesman problem has a runtime complexity of O(n!)[8] For example a problem with seven target locations would result in 56 path queries to compute the cost of traveling between each possible pair of targets assuming that the cost between two targets depends on the direction of travel. Overall there exist 7! = 5,040 tours which would need to be computed before the best tour can be identified. Thus, the path planner must provide a rapid path query method.

In summary the following requirements need to be satisfied by a suitable planning approach:

1. The path should be close to cost optimal, where the cost is computed using a metric. In this work we consider path length as the only cost of a path.
2. $\mathcal{W}$ is $\mathbb{R}^3$ and the planned path must include a height profile.
3. The path needs to be feasible with respect to the kinodynamic and nonholonomic constraints of the aircraft.
4. The path must not contain segments within $\mathcal{C}_{\text{obs}}$ or lead to ICS.
5. The planner must be capable of computing paths between multiple start and goal configurations rapidly, i.e. the planner needs to be multi-query capable.
6. The chosen approach must in principle be suitable for real time topological updates, dynamic replanning and higher level plan repairing.

Figure 1: Prometheus UA

III. Related Work

The roadmap-based path planner for the DLR ARTIS family of unmanned helicopters is presented in ref. 9. It is shown that quasi-random and lattice sampling achieve superior coverage of the workspace compared to pseudorandom sampling while the number of samples is minimized by evaluating the dispersion and discrepancy of the samples. To address motion constraints of helicopters in fast forward flight, a recursive PRM using motion primitives is suggested in ref. 10 to plan paths in changing environments. The motion primitives are generated offline and connect the samples. However, the recursive nature of the roadmap does not guarantee real-time replanning in case of large changes to the workspace, as a worst case execution time cannot be derived. To provide safe operation of the aircraft states that lead to collisions with obstacles must be avoided. Thus, inevitable collision states (ICS) are especially of interest if the world knowledge is incomplete and the aircraft relies on sensors to detect obstacles. However, it is difficult to compute ICS fast enough in order to avoid them.[6,11] Bekris lays a foundation in ref. 6 on how to compute ICS in sampling-based path planners. A real-time sampling-based planner for dynamic environments which computes paths with respect to kinodynamic constraints and its application to a simplified helicopter model was presented in ref. 12. To avoid deadends and thus ICS while modifying the search graph a safety concept is introduced which comprises a collision safe computation time. However, this planning algorithm builds on single-query planning algorithms and is not well-suited to problems which require multiple path queries. Recently, the basic PRM and RRT algorithm were proven to be suboptimal.[13] However, provably optimal versions of these algorithms called RRT* and PRM* were presented, for which the solution converges asymptotically to the optimal solution. These methods require a steering function to connect nodes, which might invoke numerical trajectory optimization for systems with dynamics.[14] Obermeyer presents a roadmap-based path planner for fixed-wing UAV for intelligence, surveillance and reconnaissance missions where a given number of static ground targets represented by polygons need to be visited. The UA is modeled as a Dubins vehicle and samples along the edges of the target areas are connected to obtain a roadmap.[15] The azimuth of these samples is determined using Halton sequences. As it is assumed that the aircraft travels at constant altitude, the problem is reduced to a planar planning problem. The problem of task scheduling is addressed in ref. 4 using a combination of a 3D roadmap planner using deterministic quasirandom sampling and a task assignment and task scheduling algorithm.

IV. Planning Approach

Sampling-based planning methods are favorable in higher dimensional configuration spaces with non-convex obstacle regions that can hardly be solved optimally by conventional / numerical optimization techniques. In challenging environments such as urban terrain considering the nonholonomic and kinodynamic constraints of the aircraft is critical to successful path planning, which can be achieved for example using the two approaches depicted in fig. 2

- Creating samples in $\mathcal{W}$, connecting them by straight-line segments and using post-processing to create feasible edges (fig. 2a).
- Creating samples in $\mathcal{C}$ and connecting them with an appropriate steering function (fig. 2b).

The first option yields straight-forward and fast collision checking for edges against obstacles[9] and requires the generation of samples in $\mathbb{R}^3$. However, no guarantee exists, that a collision-free post-processed solution path exists. More dynamic path segments are "retro fitted" on top of an approximated linear solution for which feasibility can only be guaranteed if the vehicle can stop. However, post-processing remains useful for approximate planning results (e.g. "raw" output from sampling based planners) in order to maximize planning resolution and thus minimize graph search time. The second option as shown in fig. 2b requires sampling in four dimensions and a steering function to connect the samples with valid paths. The first 3 dimensions are coordinates in $\mathbb{R}^3$ and the fourth dimension corresponds to an orientation, here the flight path azimuth. The fifth dimension, the flight path angle is set to zero in the sampling process. Collision checking for the resulting curves in $\mathbb{R}^3$ against the obstacles is more complicated as for the straight edges. However, the resulting edges if collision-free are guaranteed to be feasible and collision-free. Furthermore, they do not require post-processing, accelerating the path planning, once a graph of curved edges is built.

We chose the second approach, as we assume that the motion constraints will strongly influence the path in urban scenarios with narrow street canyons. Thus, a steering function for the system `x=steer(`q_{start}, q_{goal}`)`

(a) Post-Processing Approach (b) Feasible connection between samples

Figure 2: Two options to connect samples in either $\mathcal{W}$ or $\mathcal{C}$

Wingspan	3,2 m
Weight	25 kg
max. Airspeed.	55 m/s
Cruise speed	25 m/s
r_{min}	64 m

Table 1: Prometheus data

is necessary to drive the system from one configuration to another, where $\mathbf{x}$ is a trajectory in $\mathcal{W}$, i.e. a path with velocity profile. We model the aircraft as Dubins aircraft[16] to obtain such steering function.

A. Aircraft Model

The horizontal movement of the aircraft is approximated by Dubins car kinematics, i.e. a vehicle moving forward with constant velocity and a minimum turn radius. The vertical movement of the aircraft is modeled as double integrator with bounded acceleration and velocity. Using superposition analogous to ref. 12 of the shortest horizontal path[17] and the path resulting from time optimal bang-bang control for the vertical component a flight path between two positions in 3D and their prescribed orientation is computed. The shortest path for a Dubins car between an initial position and orientation to a final position and orientation consists of circular arcs and straight line segments and can be found by computing all possible 6 shortest path candidates as given in ref. 17 and choosing the shortest. All paths are either of the family "curve, straight, curve" or of the family "curve, curve, curve" depending on the relation between start and goal position and orientation. Using the superposition of the horizontal and vertical solution as described in Ref. 9 to match the execution time of the horizontal movement and vertical movement invokes iterative search over the admissible maximum acceleration of the vertical dynamics. If the execution time of the time optimal vertical control command sequence is slower than the execution time of the horizontal path, circular revolutions are added to the horizontal Dubins path.

Table 1 lists relevant data of the Prometheus UA. In particular the cruising speed of 25 m/s together with a desired bank angle of $\phi = 45$ result in a minimum turn radius of $r_{min} = v^2/(\tan(\phi) \cdot g) = 64m$. The climb rate and the acceleration bound were set to $v_{c,max} = 10m/s$ and $a_{z,max} = 20m/s^2$

B. Free-Space representation

An advantage of using a graph to represent the free configuration space in path planning is that a set of well-understood graph search algorithms exist, and that multiple target configurations can be persistently represented. For global roadmap planners, the planning process is divided into a pre-processing phase, in which the free space representation is built, and in a query phase corresponding to searches with the free space graph. The multi-query property makes roadmap-based approaches favorable in the context of scheduling multiple waypoints.

1. Sampling

An important issue to consider in the representation of the configuration space is the notion of completeness. An algorithm is considered to be complete if for any input it correctly reports whether there is a solution in a finite amount of time. In sampling-based approaches, weaker notions of completeness are tolerated. Deterministic sampling enables planners to be resolution complete, in the sense that if it is possible to solve the query at a given sampling resolution, they will provide a solution. On the other hand, the original probabilistic roadmaps[18] and other randomized variants are only probabilistically complete, that is, the probability tends to one that a solution will be found (if it exists) as the number of samples grows to infinity. The problem with this randomization of common pseudo-random roadmaps is that any machine implementation generates a sequence of pseudo-random numbers, which can lead to relatively large irregularities although a more uniform distribution is desired. Hence, the work in ref. 19 defends the use of deterministic sampling sources that can produce the effect that would theoretically be expected from uniform random sampling. Deterministic sampling with non-trivial functions like a Halton sequence[20] can be slower time-wise than pseudo-random sampling. Since the time for collision checks in the graph construction is much higher than the time to build the sampling sources, time is not a crucial runtime factor for choosing a sampling approach in this work. Such approaches can also be called quasi-random approaches.

Sampling techniques for the three position components are well understood and the different effects of pseudorandom, quasi-random and lattice based sampling are studied for example in ref. 9. In this work we rely on the quasi-random sampling using Hammersley sequences to generate the position components of the samples. Other than in ref. 9, however, a directional component is necessary to build a roadmap for the Dubins aircraft. Thus, each sample needs to include orientations, too. In this paper, we only consider the azimuth of the sample, the flight path angle is set to zero degree. Thus, we effectively search for a direction in $\mathbb{R}^2$ which can be understood as sampling $\mathbb{S}^1$.[5] The azimuth is defined to be $\chi \in [0, 2\pi)$. A deterministic sampling based on the Hammersley sequence is implemented to generate the azimuth angles of each sample as follows. The Hammersley sequence[21] for $\mathbb{R}^3$ is increased by one dimension to $(k/n,\ \Phi_{p_1}(k),\ \Phi_{p_2}(k),\ \Phi_{p_3}(k))$. By linearly mapping $\Phi_{p_3}(k) \rightarrow \chi \in [0, 2\pi)$ we implement a quasirandom sampling of the azimuth angle.

2. Graph

The generated samples are connected using the steering function based on the Dubins aircraft explained above and a graph is created. However, as the UA is a fixed-wing aircraft, it can only move forward. As the samples have a prescribed heading every node can only be left in one direction leading to a graph with only directed edges. Therefore, each pair of nodes is always connected by two directed edges with usually different cost. The roadmap is not a fully connected graph, i.e. not every node is connected to every node within the graph which avoids collision-checking especially as very long edges are more likely to collide with obstacles at low altitude. Thus, a neighborhood of nodes needs to be determined within which connection attempts between samples are made. Either considering the k-nearest neighbors[22] or introducing a neighborhood radius[23] have been shown to be most efficient:

1. **Neighborhood radius:** A node is connected to all nodes, which are within a predefined radius around it in $\mathbb{R}^3$
2. **k-Nearest:** A node is connected to its k-nearest neighbors.

In this work a neighborhood radius is chosen by default to limit the number of edges in the graph. If no connection can be made, the k-nearest approach is used with a low number for k (k$<$ 10).

Standard graph search algorithms like A*[24] or D* Lite[25] are then used to find a path on the roadmap.[5]

3. Collision Checking

During roadmap construction, every edge has to be checked for collision with the polygonal obstacle representation. Collision checking is computationally costly compared to generating samples. Since one of the requirements for this planner that is should be online capable in future versions, i.e. starting with an incomplete or inaccurate world model, which is updated in flight, we use the same methods for straight line segment collision checking and world model storage and indexing as described in ref. 9.

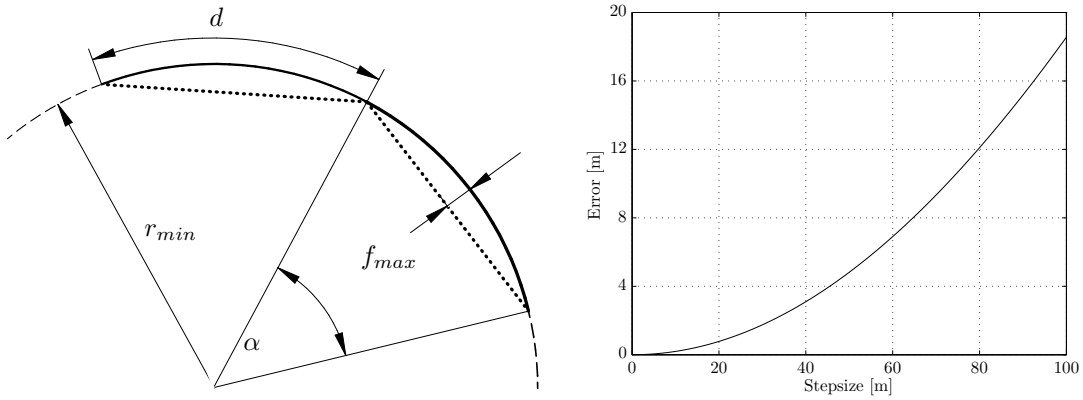
Since obstacle changes can potentially move configurations from $\mathcal{C}_{\text{obs}}$ to $\mathcal{C}_{\text{free}}$, the roadmap in this approach covers $\mathcal{C}$ with edges marked nontraversable if a collision can occur. This avoids the need for an online re-sampling of a previously unreachable volume when it is rendered free. However, this requires a full update of all affected edges costs in the roadmap if an obstacle is fed into the system.

Space partitioning can be used to avoid useless intersection tests. Voxel grids have been chosen for the space partitioning as they are fairly easy to represent, need no balancing, and allow straight forward dimensional volume expansions. As the polygons representing the obstacles are in $\mathbb{R}^3$, the voxel grid is implemented in the workspace $\mathcal{W}$.

The 3-D voxel grid is used to index polygonal objects for collision checking and graph edges that need to be updated in the case of an unforeseen obstacle pop up. This global spatial index divides the known configuration space into coarse sections. Each edge within is then checked for collision against the polygons in the same section using the method explained in ref. 26.

However, other than in the previous work the edges connecting two nodes are not straight line segments but curves in $\mathbb{R}^3$ as computed by the steering function. For collision checking, these curves are approximated by piece-wise straight line segments which are then tested for collision as described above. If one of the segments intersects with an obstacle, the whole edge is marked as untraversable.

Thus, the necessary number of collision checks depends on the discretization stepsize. Figure 3a shows the discretization of a curve segment. Each individual segment is of length d. Increasing d results in a coarser representation of the circular path but reduces the number of straight line segments, thus reducing the number of collision checks. However, this increases the maximum approximation error f_{max}.



(a) Discretization of a path segment

(b) Maximum discretization error f_{max} made for a turn with radius r_{min} using stepsize d.

Figure 3: Discretizing circular arcs with straight line segments results in an increasing approximation error with increasing length of the straight line segements.

The maximum approximation error f_{max} can be computed using $\alpha = d/r$:

$$f_{max} \quad = \quad r \cdot (1 - cos(\alpha/2)) \tag{1}$$

Figure 3b depicts the approximation error depending on the stepsize, r_{min} corresponds the minimum turning radius of the Prometheus UA.

The approximation error grows with the stepsize. Thus, when checking for collisions, this error has to be accounted for by either inflating the size of the aircraft model or the obstacles. Given a minimum turning radius $r_{min} = 64m$, the approximation error is depicted for stepsizes up to $d_{max} = 100m$ which is equivalent to $\alpha = 90°$. A stepsize of $d = 50m$ leads to an approximation error $f = 5m$. We set the stepsize to $d = 40m$ ($\alpha \approx 35°$) for the remainder of this work. This corresponds to an approximation error of $f = 3.5m$. Combining this with the wingspan of $3.2m$ and a safety margin a tunnel of 7 meter radius around each straight line path segment is required to be collision free in order for the edge to be declared collision free. Figure 4 illustrates collision checking for a roadmap node. Blue edges are collision free and red edges are in collision

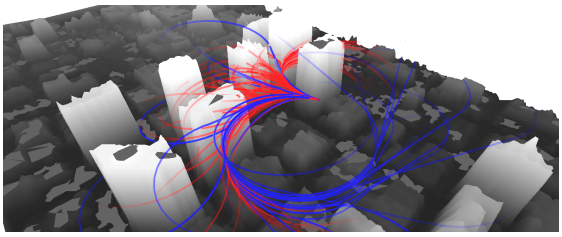


Figure 4: Red edges are outgoing edges of one node in collision with an obstacle. Blue edges are outgoing edges that are collision free.

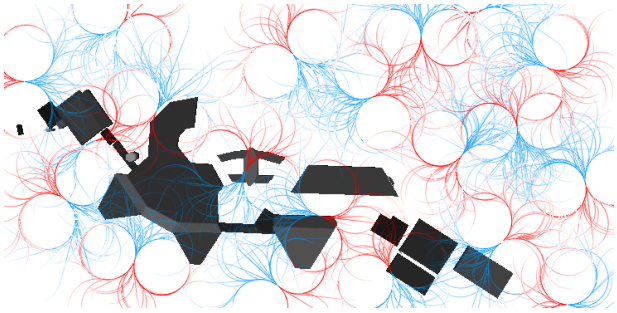


Figure 5: Illustration of a roadmap with a reduced number of nodes. All incoming edges are blue, all outgoing edges are red. The length of the edges in the figure is limited for clarity.

with at least one obstacle with respect to the criteria above and are declared untraversable during path planning.

Finally, figure 5 graphically illustrates a roadmap. Several nodes and their incoming (red) and outgoing (blue) edges are shown. Only collision free edges are drawn. To help visualize the roadmap the edges are not fully drawn but only the beginning and the end of each respective edge.

4. Inserting User Waypoints

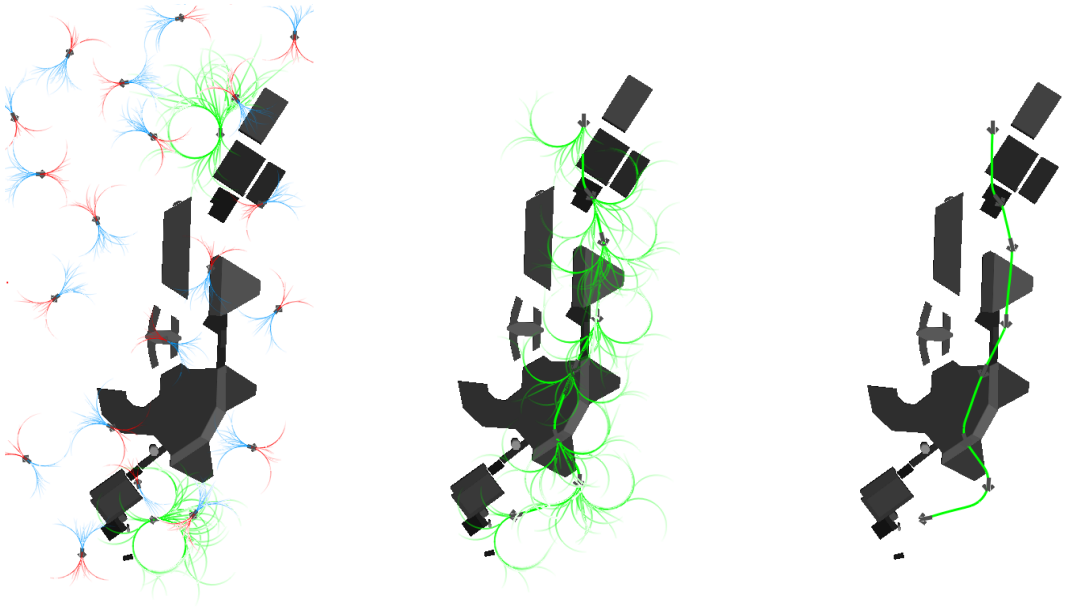
In order to search a path from an initial configuration to one or multiple final configurations using the a priori built roadmap, they need to be connected to the roadmap.

Thus, before inserting a waypoint chosen by an operator into the roadmap, there must exist outgoing edges connecting it to other nodes of the graph. A node that does not have at least one traversable outgoing edge can be considered to be an ICS. If a user waypoint is determined to be an ICS, all incoming edges are marked as untraversable. Checking if user waypoints lead to ICS can only be performed after all user waypoints have been inserted, as otherwise a newly added waypoint might change the properties of the already inserted waypoints.

Possibly, an operator might not want to specify the azimuth for each of them. Thus, two user azimuth preference are accounted for when inserting additional nodes to the roadmap. Either the operator wishes to define the path azimuth at the waypoint or the azimuth is arbitrary and only the position of the waypoint is of importance.

The use of Dubins paths, however, requires a defined azimuth for any user waypoint. Especially in challenging environments defining the azimuth for a waypoint requires consideration as the aircraft's constraint might not allow for arbitrary azimuth angles. Thus, the following approach to automated azimuth generation for user waypoints is proposed and implemented.

For any waypoint without defined azimuth, the algorithm iterates over 36 azimuths in 10° increments starting at 0°. For every iteration, the node is connected to all neighbors through incoming and outgoing edges. The azimuth that results in the largest number of collision free edges for the node is assigned to the



(a) (b) (c)

Figure 6: After inserting start (right) and goal node (left) to the roadmap (a), graph search using the roadmap (b) leads to the shortest path (c).

waypoint.

Finally, figure 6 depicts the process of finding a path between operator set start and goal using the roadmap. In (a) start position- (top of the figure) and goal (bottom) are inserted into the roadmap. The following pictures show the resulting path after search. First, the nodes belonging to the path and all their outgoing edges are shown and then the resulting path.

V. Post-processing

The previous section described the process of finding a path on a roadmap of space curves using a graph search algorithm. As the roadmap only contains connections which are feasible for the Dubins aircraft by construction, post-processing of the solution is not necessary. However, by applying post-processing, the path can be further optimized with respect to its length. Shortcut optimization as discussed below is also present in the ARTIS roadmap planner. Additionally, the ARTIS roadmap planner offers the option of spline-based interpolation of the linear path segments to provide a smooth flight path. For the fixed-wing aircraft, the added orientation dimension motivates post-processing of node orientation.

A. Shortcut optimization

As the roadmap is not fully connected, i.e. not every node is connected to every other note of the graph, possible shorter paths might exist, which can be obtained by skipping nodes which are part of the solution but not set by the operator.

During the process of short-cut optimization, an attempt to directly connect the predecessor and successor of every node which is part of the solution is made. If the connection using the Dubins aircraft yields a collision-free path, this edge is added to the roadmap.

If the cost of using the new edge is lower than the initial cost, the solution path is modified accordingly. Fig. 8 (a) illustrates the post-processing process. Three user waypoins with prescribed heading as indicated by the arrows are connected by the path indicated by the yellow solid line. The blue path depicts the improved path after short-cut optimization.

B. Path Smoothing

The azimuth of each roadmap node, that is not explicitly specified by the user, is determined by the deterministic sampling process. Thus, once a solution is found, the azimuth of each node, which is part of the solution can be modified in order to achieve a shorter path.

Fig. 7 depicts this path smoothing process. The heading of the three nodes are shown by the arrows. The original solution path (black dashed line) contains turns as the intermediate node has a prescribed azimuth that requires the turns.

The dotted line connects the first and the third node with a straight line. The angle α of this straight line connection is used to modify the azimuth of the intermediate node. The resulting path is shown by the black bold line.

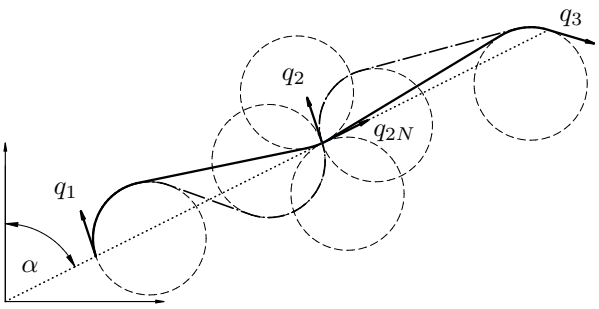
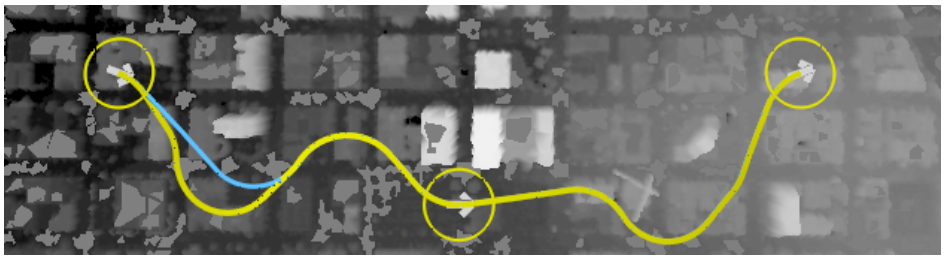
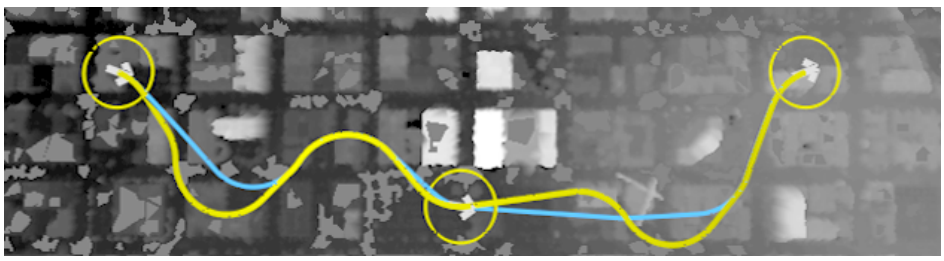


Figure 7: Smoothing of the dashed bold line; the resulting path is the bold line.

The resulting path is checked for collisions with the obstacles. Even if it is collision-free, the resulting path (q_1, q_{2N}, q_3) could be longer, than the previous path (q_1, q_2, q_3). If the resulting path is collision-free and shorter than the initial path, the solution is updated. Smoothing the whole path comprises iterating over all path segments, i.e. iterating over all (q_i, q_{i+2}). Fig. 8 (b) shows the smoothing using the previous example.

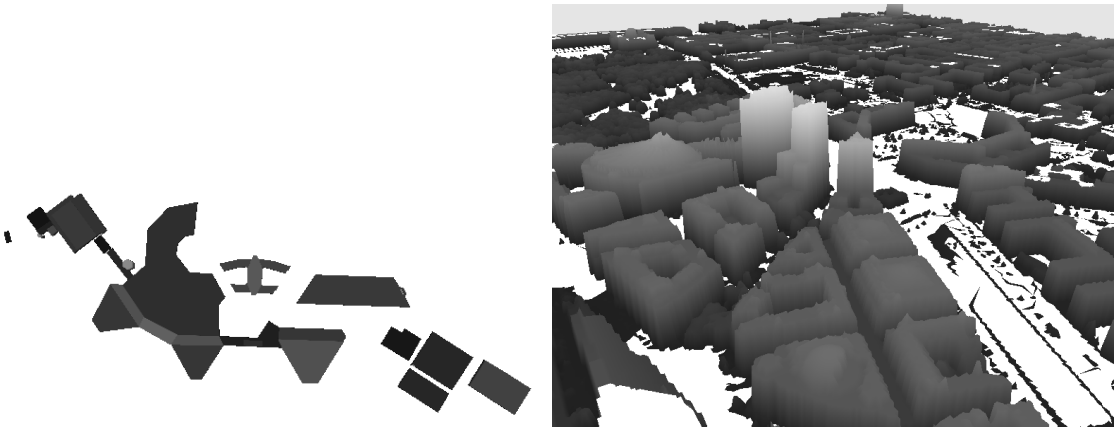


(a) Shortcut optimization



(b) Azimuth smoothing

Figure 8: Post-processing of an initial solution.



(a) (b)

Figure 9: (a) shows a sparse test environment with few large obstacles. (b) shows the urban scenario (Berlin city) obtained using remote sensing resulting in a digital elevation map to create a polygonal world model

VI. Optimization of Roadmap-Parameters

The results obtained from the this planning approach depend on the choice of several parameters such as the distance between samples and the sampling strategy. The following section covers the choice of these parameters based on test scenarios.

The roadmap planner should not be optimized towards one special scenario. It should rather be applicable and efficient for a wide range of scenarios. Thus, finding a set of parameters that allow different environment types is the main focus of this section.

A. Test scenarios

To evaluate the planner, we use two scenarios, where one is sparsely populated with large obstacles, and the other one is an urban scenario which is densely populated with obstacles and represented the center of downtown Berlin in Germany. Both are shown in fig. 9. Several tall building are higher than most of the other building such that direct connections between waypoints are not possible even above the roofs of most buildings. All measurements were performed on a Pentium 4 desktop system with 3Ghz and 2GB RAM. The planner is implemented using C++, compiled with the VC8 compiler and executed under Microsoft Windows XP 32bit.

B. Sampling density

In sparse terrain a low sampling density is sufficient to compute solutions for most path queries. However, urban terrain, i.e. environments with a high obstacle density require a higher sampling density due to the more challenging configuration space. However, increasing the sampling density requires more computation time and memory. Thus, a trade-off needs to be found.

Due to the dynamic constraints of the aircraft, introducing a minimum Euclidean distance between two samples is beneficial. Even if connecting very close nodes is possible, the benefit of adding these edges to the roadmap is outweighed by the cost of computation and storage as these edges are rather long due to the necessary turns.

C. Sampling strategy - Neighborhood definition

As mentioned before, the chosen roadmap is not a fully connected graph. The presented planning approach uses a neighborhood radius to determine which nodes to connect. The following points favor shorter edges, i.e. a smaller neighborhood radius. Collision checking for long edges is computationally expensive.[5] As each edge is approximated using straight line segments multiple collision checks need to be performed for longer

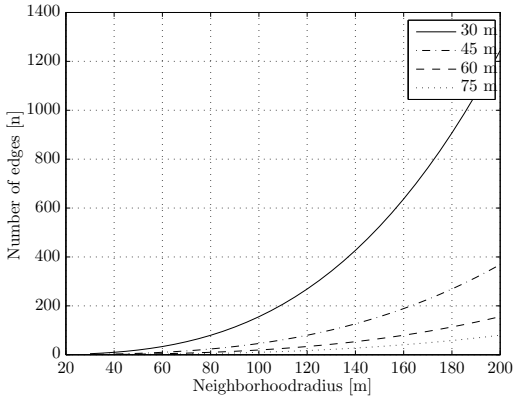
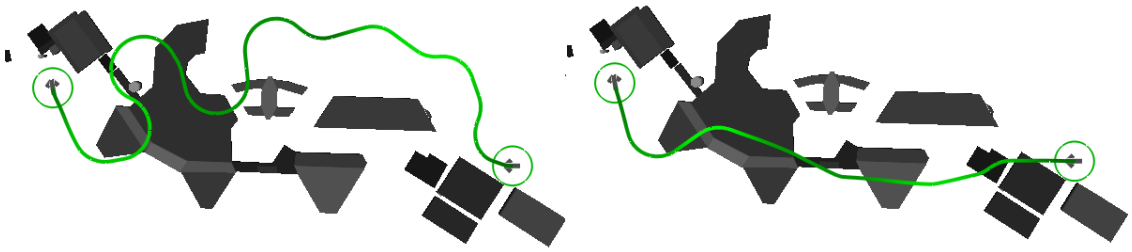


Figure 10: Number of edges for a node depending on the neighborhood radius and the distance between samples.



(a) (b)

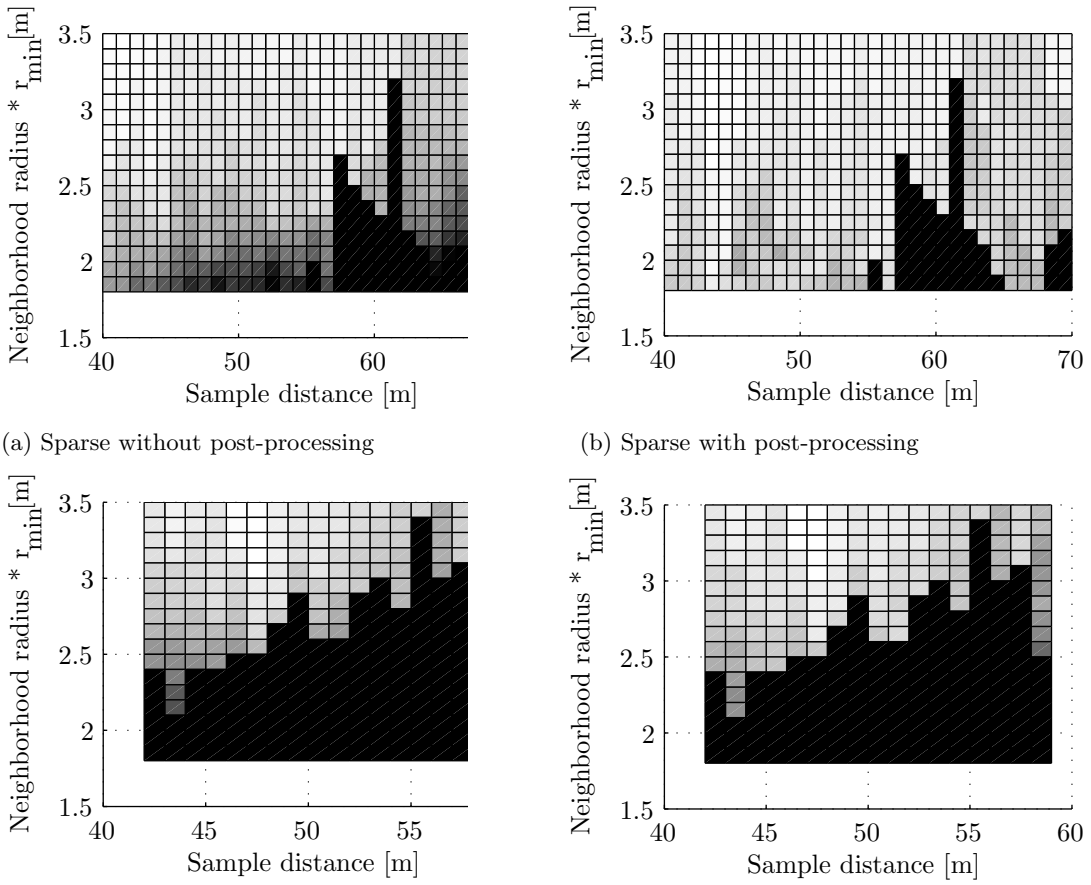
Figure 11: Non post-processed paths with neighborhood radius $2,4 \cdot r_{min}$ (a) and $3 \cdot r_{min}$ (b) for a sample distance of 50 m

edges. Furthermore, longer edges are more likely to intersect with obstacles. Thus, very long edges should be avoided. Figure 10 shows the number of neighbors of a sample depending on the neighborhood radius for different numbers of samples. The number of neighbors grows exponentially with the radius. Especially for densely sampled roadmaps, the neighborhood radius should be chosen small.

However, the neighborhood radius must at least be equivalent to the minimum distance between samples as determined by the sampling density. Furthermore, paths resulting from roadmaps with very short edges tend to exhibit multiple unnecessary turns as due to the azimuth sampling, nodes which are close to each other with respect to the Euclidean distance in $\mathbb{R}^3$ might have different azimuth angles. Thus, increasing the neighborhood radius typically yields shorter solution paths. Fig. 11 shows this behavior, where the same path query is solved using roadmaps with two different neighborhood radii without post-processing. While for $2,4 \cdot r_{min}$ the resulting path contains multiple turns, a larger neighborhood radius of $3 \cdot r_{min}$ yields a much straighter path. The choice of an appropriate value for the neighborhood radius based on simulations will be discussed in a later section.

VII. Simulation

The following simulations were conducted in the sparse environment as well as in Berlin downtown to guide the choice of the aforementioned roadmap parameters and to illustrate the planners applicability.



(a) Sparse without post-processing

(b) Sparse with post-processing

(c) Berlin without post-processing

(d) Berlin with post-processing

Figure 12: Path cost for different sampling distances and neighborhood radii for the sparse and the Berlin downtown scenario.

A. Parameter Variation

To determine the sampling density and neighborhood radius, simulations are carried out using varying parameter sets to determine a feasible combination. The sampling density and the neighborhood radius are varied and the costs of the resulting paths with and without post-processing are compared. The results are depicted in fig. 12. The neighborhood radius is given in multiples of r_{min}. Each cell in the figure corresponds to one set of parameters. The cost is color-coded using different shades of grey, where lighter shades correspond to lower costs. Black cells refer to parameter setting for which at least one user waypoint cannot be reached using the roadmap.

The figure shows that the solution quality decreases with increasing distance between samples and decreasing neighborhood radius. Additionally, the scenario "Berlin", i.e. the urban scenario seems to be more difficult than the sparse scenario. The following section covers this aspect in more detail.

For the sparse scenario, a sampling distance of approximately 55 m and a neighborhood radius of $2,5 \cdot r_{min}$ lead to good results. The post-processing has a significant influence on the overall length of the path. Even poor initial solutions benefit significantly from post-processing. As most of the obstacles have the same height, the shortcut optimization is very efficient at altitudes above the roofs of the buildings.

For the denser urban scenario, sparse sampling leads to planning failure as can be observed by the black cells in the figure. A neighborhood radius smaller than $2,5 \cdot r_{min}$ also leads to bad results. Results start to improve for radii equal or larger than $3 \cdot r_{min}$. However, the distance between samples should not exceed 50 m. Post-processing does not have such a strong effect on the path as it had in the previous scenario. The environment contains several tall building which prohibit significant shortcut optimization.

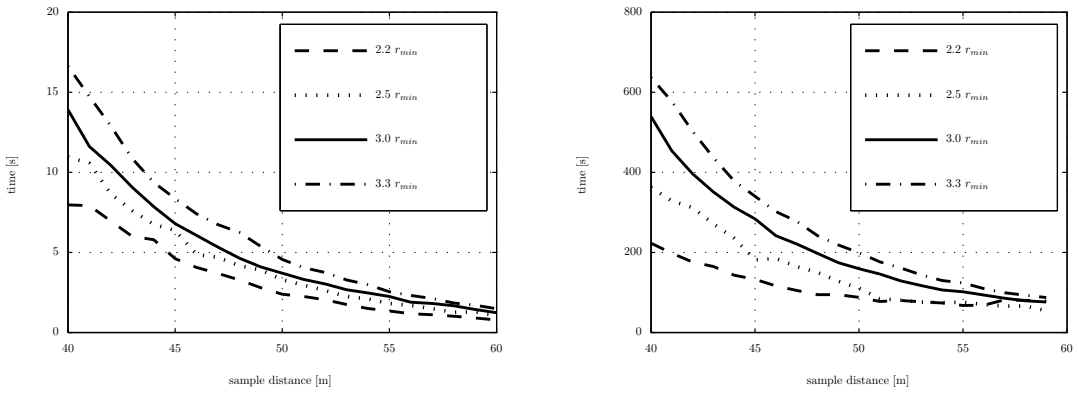


Figure 13: Roadmap construction time for the sparse scenario (a) and downtown Berlin (b).

Fig. 13 shows the roadmap construction time for different sampling densities and neighborhood radii. The roadmap for the sparse scenario (left) requires a lower construction time than the scenario Berlin, as its dimensions are much smaller. A neighborhood radius $3,5 \cdot r_{min}$ leads to a initial roadmap construction time of 600 seconds for the urban scenario. However, as discussed before, a sample distance of 50 m and a neighborhood radius of $3 \cdot r_{min}$ already yield short paths. This parameter setting requires a shorter computation time of only 180 seconds.

Finding a balance between good coverage of the free space and computation time is difficult. Figure 14 shows the number of edges within the roadmap for which the horizontal path is of type CCC over the neighborhood radius. For neighborhood radii smaller than twice the minimal turning radius, one third of all connections is of type CCC which is in general not desirable, as these edges tend to be long and collision-prone. For neighborhood radii greater and above $2,8 \cdot r_{min}$ good solutions even without post-processing are achievable. For the urban Berlin scenario the distance between samples cannot be greater than 50 m to obtain good results. Increasing the neighborhood radius to more than $3 \cdot r_{min}$ increases the computation time excessively. Thus, a distance between samples of 50 m and a neighborhood radius of $3 \cdot r_{min}$ yields the best trade-off for the two scenarios evaluated.

B. Excluding paths of type CCC from the roadmap

As explained previously, time optimal paths for a Dubins vehicle are either of type CSC or CCC. Paths of type CCC are typically only time-optimal, if the Euclidean distance between start and goal configuration is very small and there is a significant difference in azimuths. However, the path length of a CCC path is usually long compared to the Euclidean distance between the two configurations. As the objective of the path planner is to find a short path, CCC paths are usually not part of a solution paths. This raises the question, whether CCC tpye paths should not be considered during roadmap construction. As shown above, the percentage of CCC type connections in the roadmap for reasonable neighborhood-radii is small. Thus, excluding CCC paths from the roadmap only reduces the number of total edges in the graph by a small fraction for the neighborhood radius determined above.

For neighborhood radius of approximately $3 \cdot r_{min}$ fig. 15 shows the time savings due to the reduced number of collision checks if CCC edges are excluded from the roadmap during roadmap construction. Filtering CCC edges reduces the computation time by 13 % and 17 % respectively.

However, filtering these edges might results in nodes being disconnected from the graph which can only be connected to the roadmap by means of collision-free CCC paths. Thus, in a worst-case scenario the possible sole solution path to a user defined waypoint might be filtered. Thus, the filtering is optional in our implementation.

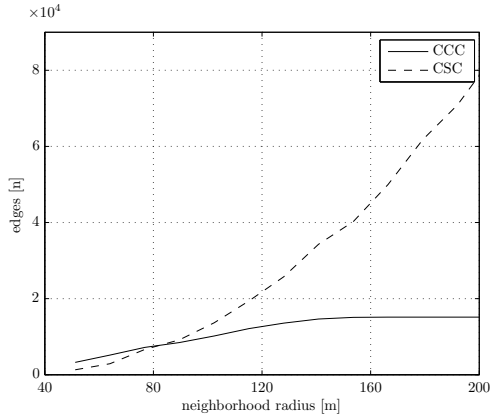


Figure 14: Number of CSC and CCC edges within the roadmap for a sampling distance of $45m$ in the sparse scenario.

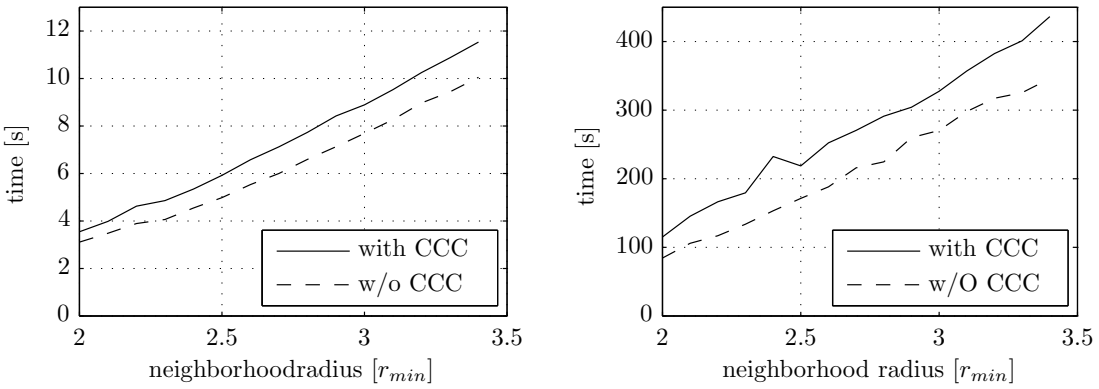


Figure 15: Roadmap construction time for the sparse scenario (a) and Berlin downtown (b).

C. Resolution of the world representation

As explained before, the terrain is represented using a polygonal world representation. However, there is a trade-off between necessary storage, computation time spend collision checking against each polygon within a certain volume and the accuracy of the world model.

Depending on the required precision the resolution of the world model can be adapted. As the roadmap can be understood as free-space representation, the resolution of the world model has to be higher than the sampling density. Furthermore, the chosen resolution also depends on the the terrain itself. Especially urban terrain requires a high resolution to represent narrow street canyons with sufficient accuracy. As computation and storage increases significantly with resolution, we use the mesh simplification procedure presented in ref. 27

However, for unmanned aircraft with limited maneuverability a world representation at a very high resolution is not efficient as modeling for example narrow passages that cannot be entered by the vehicle due to its constraints could be omitted from the world model.

We used the Berlin downtown scenario to evaluate different resolutions. To force the path to enter narrow passages, waypoints were placed accordingly, see fig. 16.

Table 2 lists memory footprint of the world model for the original model, the simplified mesh and whether the planning succeeded for terrain representations with a resolution of 1 m, 2 m, 4 m and 8. A safety distance of 5 m was used. Only the world representations at 1 m and 2 m resolution lead to an overall successful plan. For 4 m and 8 m not all user waypoints could be reached.

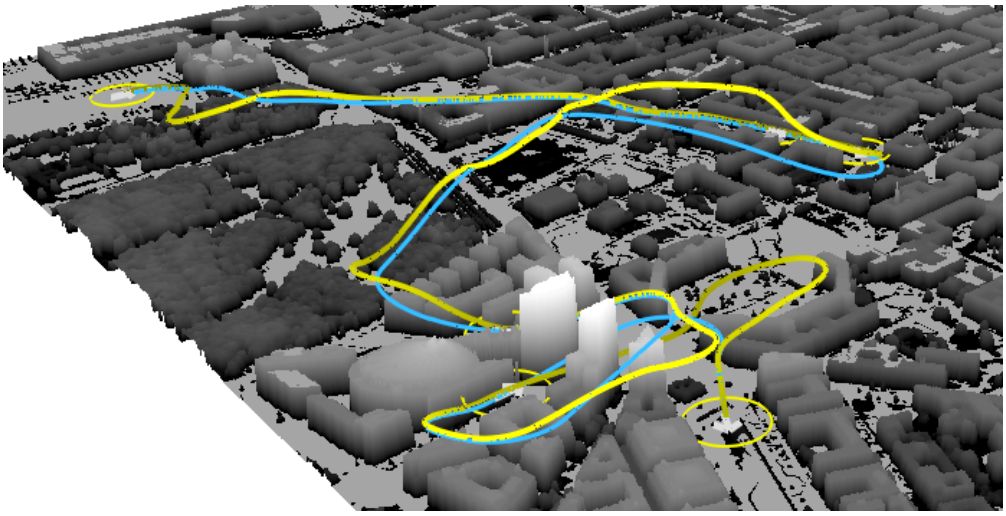


Figure 16: Path before (yellow) and after post-processing (blue) through multiple waypoints close to the ground in downtown Berlin.

x/y-Res. [m]	Size. [MB]	Size. red. [MB]	RM [s]	Goal reached
1	729	276	611	Y
2	181	99	286	Y
4	45	32	191	N

Table 2: Comparison of different world resolutions

Table 2 depicts the storage footprint of the world model with and without the aforementioned mesh simplification. Roadmap construction was only performed for the simplified meshes. Reducing the resolution from 1 m to 4 m reduces the memory requirement by a factor of 8. The roadmap construction is sped up by a factor of 3.

A high resolution is in general beneficial for complex urban terrains. For the given planner and vehicle, a resolution of 2 m is found to be suitable, which allows for optimal use of the aircraft's capabilities. A higher resolution increases the computation time without improving the results. The sampling density needs to be chosen accordingly in order to leverage the high world model resolution.

Finally, fig. 16 shows the paths obtained for the example urban scenario inlcuding post-processing using the parameters determined above.

VIII. Conclusion

This paper presented the development of a path planner for the Prometheus fixed-wing aircraft operating in environments densely populated by obstacles. We use a Dubins aircraft model to account for the dynamic and nonholonmic constraints of the aircraft and build a roadmap using deterministic sampling in position and azimuth to generate roadmap nodes. Optional geometric post-processing steps improve the found paths with respect to its length. A good set of parameters for the roadmap including sampling density and neighborhood radius is obtained through simulation studies of varying parameter sets. The planner solves path planning queries quickly even in challenging environments and produces intuitive and short paths which are feasible with respect to the constraints of the Dubins aircraft. Most computation is done once before any path query is performed. Thus multiple path queries can be answered rapidly. This helps solving task scheduling problems with real flight path length information obtained from the roadmap planner. Future research could consider smoothing the flight path angle or adding the flight path angle as a degree of freedom in the sampling process. Additionally, the online capabilities for pop-up obstacles as presented in ref. 9 could be examined

as the current implementation brings the necessary foundation but has not yet been made completely online capable.

Acknowledgments

The authors would like to thank the team of the DLR Institute of Flight Systems - Unmanned Aircraft Department for their support of this work.

References

[1]Niendorf, M., Adolf, F., Gerhard, T., *Behavior-Bases Onboard Mission Management for an Unmanned Fixed-Wing Aircraft*, in: AIAA Infotech@Aerospace, Garden Grove, USA 2012.

[2]Adolf, F.-M., *Evaluation of the ARTIS sampling-based path planner using an obstacle field navigation benchmark*, AHS Forum 68, Forth Worth, TX, 2012.

[3]Adolf, F.M., Hussein, M. A., Goerzen, C., *Trajectory Time Reduction using Field of View-based Smoothing of Roadmap-based Paths*, AHS Forum 69, Phoenix, AZ, 2013.

[4]Adolf, F., Andert, F., Lorenz, S., Goormann, L., Dittrich, J., *An Unmanned Helicopter for Autonomous Flights in Urban Terrain*, in: German Workshop on Robotics, Berlin (Springer) 2009, pp. 275–285.

[5]LaValle, S. M., *Panning Algorithms*, Cambridge University Press, 2006.

[6]Bekris, E. B., *Avoiding Inevitable Collision States: Safety and Computational Efficiency in Replanning with Sampling-based Algorithms*, in: International Conference on Robotics and Automation, Anchorage, USA 2010.

[7]Canny, J. F. and Reif, J. H., *Lower Bounds for Shortest Path and Related Problems*, IEEE Symposium on Foundations of Computer Science, Los Angeles, CA, 1987, pp. 49-60.

[8]Applegate, D. L., Bixby, R. E., Chvatal, V., and Cook, W. J., *The Traveling Salesman Problem: A Computational Study*, Princeton Series in Applied Mathematics, Princeton University Press, Princeton, NJ, USA, 2007.

[9]Adolf, F., Andert, F., *Rapid Multi-Query Path Planning for a Vertical Take-Off and Landing Unmanned Aerial Vehicle*, in: AIAA Journal of Aerospace Computing, Information, and Communication, 8(11), 2011, pp. 310–327.

[10]Frewen, T. A., Sane, H., Kobilarov, M., Bajekal,S., Chevva, K. R., *Adaptive Path Planning in a Dynamic Environment using a Receding Horizon Probabilistic Roadmap*, in: Experimental Demonstration, AHS Specialists' Meeting on Unmanned Rotorcraft, Tempe, USA 2011.

[11]Bouraine, S., Fraichard, T., Salhi, H., *Provably safe navigation for mobile robots with limited field-of-views in dynamic environments*, in: Autonomous Robots, 32(3), 2012, pp. 267–283.

[12]Frazzoli, E., Dahleh, M. A., Feron, E., *Real-Time Motion Planning for Agile Autonomous Vehicles*, in: AIAA Journal of Guidance, Control, and Dynamics, 25(1), 2002, pp. 116–129.

[13]Karaman, S., Frazzoli, E., *Sampling-based Algorithms for Optimal Motion Planning*, in: International Journal of Robotics Research, 30(7), 2011, pp. 846–894.

[14]Jeong H. J., Karaman, S., Frazzoli, E., *Anytime computation of time-optimal off-road vehicle maneuvers using the RRT**, 2011 50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC), 12-15 Dec. 2011, pp. 3276–3282.

[15]Obermeyer, K. J., Oberlin, P., Darbha, S., *Sampling-Based Roadmap Methods for a Visual Reconnaissance UAV*, in: AIAA Guidance, Navigation, and Control Conference, Toronto, Canada 2010.

[16]Chitsaz, H., and LaValle, S.M., *Time-optimal paths for a Dubins airplane." Decision and Control*, 2007 46th IEEE Conference on. IEEE, 2007.

[17]Dubins, L. E., *On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents*, in: American Journal of Mathematics, 79(3), 1957, pp. 497–516.

[18]Vestka, L. E. K. P., Claude Latombe, J., and Overmars, M. H. *Probabilistic Roadmaps for Path Planning in Highdimensional Configuration Spaces*, IEEE Transactions on Robotics and Automation, Vol. 12, 1996, pp. 566-580.

[19]Branicky, M. S., LaValle, S. M., Olson, K., and Yang, L., *Quasi-Randomized Path Planning*, Proceedings of the IEEE International Conference on Robotics and Automation, Seoul, Korea, 2001, pp. 1481-1487.

[20]Halton, J., *On the Efficiency of Certain Quasi-random Sequences of Points in Evaluating Multidimensional Integrals*, Numerische Mathematik, Vol. 2, 1960, pp. 84-90.

[21]Wong, T., Luk, W., Heng, P., *Sampling with Hammersley and Halton Points*, in: Journal of Graphics Tools , 2(2), 1997, pp. 9–24.

[22]Geraerts, R., Overmars, M. H., *A Comparative Study of Probabilistic Roadmap Planners*, in: Workshop on Algorithmic Foundations of Robotics, Nice, France, 2002, pp. 43–58.

[23]Geraerts, R.J., *Sampling-based Motion Planning: Analysis and Path Quality*, Dissertation, Utrecht University, 2006.

[24]Hart, P. E., Nilsson, N. J., and Raphael, B., *A Formal Basis for the Heuristic Determination of Minimum Cost Paths*, IEEE Transactions on Systems Science and Cybernetic, Vol. 4, No. 2, 1968, pp. 100-107.

[25]Koenig, S., Likhachev, M., *D* Lite*, in: AAAI Conference of Artificial Intelligence, Edmonton, Canada 2002, pp. 476–483.

[26]Bergen, G., *Collision Detection in Interactive 3D Environments*, Morgan Kaufmann Publishers, 2004.

[27]Adolf, F., Hirschmüller, M., *Meshing and Simplification of High Resolution Urban Surface Data for UAV Path Planning*, in: Journal of Intelligent & Robotic Systems, 61(1-4), 2011, pp. 169–180.