Flight Planning Using Time Annotated B-Splines for Safe Airspace Integration

Nicolai Voget, Johannes Krimphove and Dieter Moormann

Abstract Within the near future a high rise of automatic flights of unmanned aerial vehicles beyond visual line of sight is expected. In order to efficiently coordinate these flights, unmanned traffic management systems will require precise information about the planned trajectories. This paper introduces an approach for using time annotated non-uniform B-Splines to represent the 4D trajectory of a UAV. First, it depicts the mathematical requirements for spline based representations that evolve from the dynamic constraints of hybrid aircraft, i.e. UAVs that are able to hover as well as to fly airborne at high speed. We then present a 4D representation by using time as primary parameter of splines. This imposes additional restrictions on the modeling of the used splines. In this paper we show how to fulfill these restrictions for various cases, such as straight line segments under a given acceleration or turns at constant airspeed.

1 Introduction

Until recently, automatic flights of unmanned aerial vehicles (UAVs) beyond visual line of sight (BVLOS) have been highly restricted, resulting in very few BVLOS operations. However, recent changes in aviation regulations in combination with current work of the Joint Authorities for Rulemaking on Unmanned Systems (JARUS), especially SORA [6], lead the way for permission procedures of routine UAV operations even outside the operator's line of sight. As this is expected to significantly increase the number of unmanned flights, sufficient technical means have to be developed in order to avoid collisions and efficiently coordinate these flights.

Different stakeholders have published blueprints about their understanding of an unmanned traffic management system (UTM) [1, 13]. A common point in these is

Nicolai Voget
Institute of Flight System Dynamics, RWTH Aachen University, e-mail: voget@fsd.rwth-aachen.de

the need for 4D trajectories, i.e. a representation of the UAV's path over time. The usage of 4D trajectories allows smaller separation limits compared to the current ATM situation where information is only available about the aircraft's path itself, i.e. the 3D representation. This results in bigger uncertainty of collision predictions as the future trajectory of the aircraft has to be estimated based on its current speed. To account for this uncertainty, ATMs have to apply high separation limits.

It has to be expected that unmanned aircraft operated beyond visual line of sight will be required to submit a trajectory for the entire flight, i.e. until landing, for approval to the UTM. The reason why trajectories have to be planned until landing is obvious: Assumed aircraft would plan their trajectories for a part of the remaining flight only, they could enter a state in which a safe flight to the desired landing site can no longer be performed. This can happen if e.g. a section of the remaining path that had not yet been provided to the UTM has been reserved for another UAV in the meantime. The only way to prevent this would be to reserve the corridor to the landing site for a superfluously large amount of time. By providing the entire trajectory even before start, the UTM can ensure a safe flight without the need for trajectory replanning in-flight, apart from reacting to other intruders not communicating with the UTM.

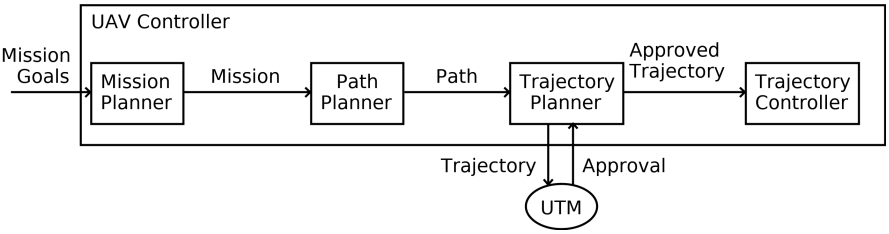

Fig. 1 UAV Controller Cascade

To enable UAVs to perform reasonable trajectory planning, we propose the following general layout of a UAV controller, which has been adapted from [2], see Figure 1. The cascade is input abstract mission goals such as examination of an area or hovering over a specific point of interest. This set of goals is then sorted by a mission planner, possibly omitting some of the goals, e.g. due to insufficient remaining energy. Next, a path is calculated that connects and fulfills the given mission goals. This path consists of straight line segments, circular turn segments and hover phases. Additional to the 3D geometry, each element is annotated with a desired speed, or duration in case of hovering. The trajectory planner then generates a 4D trajectory for this path that allows for the UAV's dynamic constraints, e.g. its maximum acceleration. This generated trajectory gets transmitted to the UTM where it is checked for collisions with other trajectories that have already been registered. It is only after after a successful UTM approval that the trajectory gets passed on to the trajectory controller.

Nevertheless, even assuming a feasible trajectory has been determined and approved by the UTM, various reasons can render a recalculation necessary, including

a change of mission goals, detection of previously unknown obstacles or change in wind conditions causing the trajectory to break the UAV's dynamic constraints. While an update of mission goals doesn't obligatory require an immediate recalculation of the trajectory as it still guarantees a safe flight, the other two circumstances most certainly do. Therefore, a deterministic approach has to be chosen for the trajectory planner in order to guarantee definite computation time limits.

Existing work on 4D trajectory planning mainly focused on waypoint based approaches [3, 5]. For this approach a sequence of time—position pairs with strictly increasing time values is calculated based on a given path. While this enables easy determination of the UAV's position for any point in time included in the sequence, the exact position for any other point in time depends on the exact implementation of the UAV's controller. Therefore, the interval between two consecutive points in time in the sequence has to be relatively small to prevent high uncertainties. This in turn leads to an inefficiently high number of waypoints.

Additionally, the determination of other details of the trajectory can become a complex problem. For example, the exact velocity at a given point isn't unambiguously defined by the progress of waypoints. The most simple approach to determine the correct velocity would be to assign to each waypoint the velocity vector obtained from dividing the spatial distance to the next waypoint by the respective time interval. However, this would lead to piece-wise constant velocities with jumps at the transition from one segment to the next. As this would require infinite accelerations, this approach is obviously not feasible. Therefore, a trajectory controller has to implement a quite complex functionality in order to generate a steady velocity curve that is consistent with the given trajectory.

For 3D applications, B-Splines have been used to continuously represent a desired path [8, 9]. As B-Splines of degree p create piece-wise defined, p times differentiable polynomials, smooth acceleration curves can be obtained easily. Nevertheless, the exact position after a given time or distance cannot be determined analytical. Therefore, different approaches have been followed in order to generate continuous 4D trajectories. In order to coordinate cooperating UAVs [4] introduces an additional mapping which translates a point in mission time to the corresponding curve parameter. However, for this approach Pythogorean Hodograph Bézier curves have been used to represent the individual paths of the cooperating UAVs. This leads to redundant information compared to the use of B-Splines as the boundary conditions at the transition between trajectory segments have to be represented by parameters of both segments, thus increasing the complexity of the optimization problem. In contrast, time annotated B-Splines have been used in [7] to precisely describe a look-at trajectory for a camera shot as well as the corresponding trajectory of the quadrocopter to which the camera is attached.

However, all of the mentioned approaches require the solution of an optimization problem in order to determine the desired trajectory. As it is impossible to guarantee feasible results after a given time for these algorithms we will present a deterministic approach for trajectory calculation assuming a previously planned path is available.

The remainder of this paper is structured as follows: First, we will analyze the dynamic constraints that arise from flight mechanics of unmanned aircraft. Consid-

ering these constraints, we introduce a mathematical representation of 4D trajectories based on non-uniformal B-Splines. We will then present how the convex hull property of B-Splines can be used to implement an efficient algorithm that checks for collisions of two given trajectories. The main part of this paper consists of our trajectory planning approach which allows to deterministically calculate an appropriate B-Spline for an input path.

2 Dynamic Constraints of Aircraft Trajectories

For a trajectory representation to be able to accurately reproduce the trajectory of an aircraft, it has to obey the relevant dynamic constraints arising from the aircraft's flight mechanics. In order to provide a representation suitable for trajectories of any kind of UAV, we will perform this analysis on tiltwing aircraft: A tiltwing aircraft is a hybrid aircraft which is able to hover like a multirotor as well as to fly wing-borne during cruise flight [12]. The transition between those two flight states is achieved by rotating its wing around the aircraft's lateral axis: During cruise flight, the aircraft resembles a fixed-wing aircraft resulting in lift being generated by the wing while the main propulsion system is used to compensate for the aircraft's drag. In order to decrease the desired airspeed, the wing is slowly rotated until the main propellers' thrust points upwards. At this point, thrust is used to counter the gravitational force and no force is applied in horizontal direction, so the aircraft hovers with an airspeed $V_a = 0$.

During transition between cruise and hover flight, the rotation of the main wing also alters the direction of the moments of torque produced by the aircraft's ailerons as well as by differential thrust. While in cruise flight the ailerons produce a moment around the roll axis and differential thrust creates a yaw moment, this attribution is swapped during transition resulting in differential thrust producing a roll moment whereas the ailerons are used to produce a yaw moment in hover flight. Additionally, as the horizontal airspeed decreases during transition, so does the aileron's effectiveness. Therefore, a tail propeller is needed to stabilize the aircraft around the pitch axis during low speed flight.

As the thrust vector always lies inside the aircraft's symmetry plane, no direct force can be applied in lateral direction. Instead, the aircraft has to tilt its lift vector by rolling. Even assuming negligible short time constants for control surface deflections, a specific bank angle cannot be attained immediately as the roll rate is limited by various factors. Additionally, in hover flight no direct force can be applied in forward direction as the thrust points upwards. Therefore, the aircraft has to pitch in order to produce a longitudinal acceleration. Again, even assuming instantaneous change of the tail propeller's thrust, having a limited maximum pitch rate prevents abrupt changes of longitudinal acceleration during low-speed flight.

3 4D Trajectory Representation

With the aircraft's acceleration being the second derivative of its position with respect to time, requiring steadiness for acceleration implies the need for the aircraft's trajectory being three times continuously differentiable. One way to represent three times differentiable functions is by using B-Splines of third degree.

3.1 Non-Uniformal B-Splines

A B-Spline of degree p with k segments is defined by a knot vector T and a set of control points P:

$$T = \left\{ \tau_0 = \ldots = \tau_p, \tau_{p+1}, \ldots, \tau_{p+k} = \ldots = \tau_{2p+k} \in \mathbb{N} \right\} \tag{1}$$

$$P = \left\{ \mathbf{P}_i \in \mathbb{R}^3, 0 \leq i \leq p+k-1 \right\} \tag{2}$$

The knots have to be monotonically increasing, i.e. $\tau_i \leq \tau_{i+1}$, with the additional restriction $\tau_i < \tau_{i+p+1}$.

Using the knot vector, so called B-Spline basis functions $N_i^j(u)$ can be defined:

$$N_i^0(u) = \begin{cases} 1 & \text{for } \tau_i \leq u < \tau_{i+1} \\ 0 & \text{else} \end{cases} \tag{3}$$

$$N_i^j(u) = \frac{u - \tau_i}{\tau_{i+j} - \tau_i} N_i^{j-1}(u) + \frac{\tau_{i+j+1} - u}{\tau_{i+j+1} - \tau_{i+1}} N_{i+1}^{j-1}(u) \tag{4}$$

The position can then be calculated by weighting each control point with its respective basis function:

$$\mathbf{x}(u) = \sum_{i=0}^{p+k-1} N_i^p(u) \mathbf{P}_i \,. \tag{5}$$

Looking at (4) some properties of B-Splines can be seen [11]:

1. The $N_i^j(u)$ form piece-wise polynomials of degree j. Therefore, $\mathbf{x}(u)$ can also be represented by piece-wise defined polynomials of degree p.
2. $N_i^j(u)$ is non-zero only for $\tau_i \leq u < \tau_{i+j+1}$. Together with (5) this implies that changing $\mathbf{P}_i$ impacts the spline in this parameter range only. This property is called *local support* of the control points.
3. It can be proven that $0 \leq N_i^j(u) \leq 1$ and $\sum_{i=0}^{2p+k-j-1} N_i^j(u) = 1$. Therefore, each point $\mathbf{x}(u)$ lies inside the convex hull for the $p+1$ control points $\mathbf{P}_i$, $\tau_{i-p} \leq u < \tau_{i+1}$. Figure 2 displays a B-Spline of third degree with five segments illustrated by different line styles. Also, the corresponding control points and the convex hull for all control points effective during the second segment are drawn.

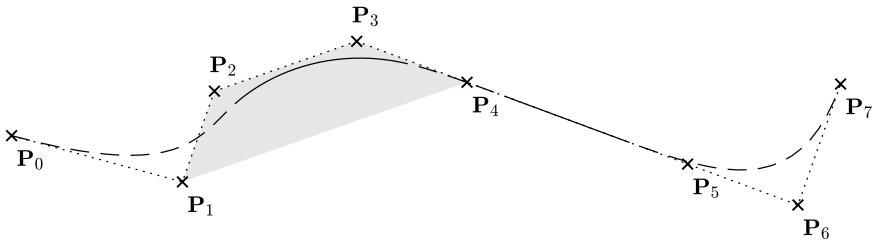

Fig. 2 Third Degree B-Spline Consisting of Five Segments Displayed With Control Points and Convex Hull for Second Segment.

3.2 Time Annotated B-Splines

Using the derivatives

$$N'^{j}_{i}(u) = \frac{p}{\tau_{i+j} - \tau_i} N^{j-1}_{i}(u) - \frac{p}{\tau_{i+j+1} - \tau_{i+1}} N^{j-1}_{i+1}(u) \quad \text{and} \tag{6}$$

$$N''^{j}_{i}(u) = \frac{p}{\tau_{i+j} - \tau_i} N'^{j-1}_{i}(u) - \frac{p}{\tau_{i+j+1} - \tau_{i+1}} N'^{j-1}_{i+1}(u) \tag{7}$$

the (non-normalized) vectors denoting the tangent and curvature at $\mathbf{x}(u)$ can be calculated by

$$\mathbf{x}'(u) = \sum_{i=0}^{p+k-1} N'^{p}_{i}(u)\,\mathbf{P}_i\,. \tag{8}$$

$$\mathbf{x}''(u) = \sum_{i=0}^{p+k-1} N''^{p}_{i}(u)\,\mathbf{P}_i\,. \tag{9}$$

Multiplying these with $\mathrm{d}u/\mathrm{d}t$ and $\mathrm{d}^2u/\mathrm{d}t^2$, respectively, one actually gets correct velocity and acceleration vectors. However, $u(t)$ cannot easily be represented by an analytic function in general. Different approaches have therefore been proposed in order to link position and time of a given B-Spline based path.

Reference [8] assumes a constant speed of the controlled UAV and uses the curvature of the spline to determine the appropriate lateral acceleration. For this approach a feasible path has to be assumed, i.e. the maximum allowed curvature is constant for the whole path, depending only on the desired constant speed. In contrast, [9] presented a controller concept for an unmanned helicopter. With helicopters being able to decelerate even to a full stop in air, this approach determines the current velocity command by analyzing the upcoming speed limits due to high curvatures.

However, in both cases the actual position cannot be determined efficiently for a given point in time: Even using the constant speed approach, the entire trajectory has to be integrated in order to identify the position for a given traveled distance or

interval. Considering the second approach where the speed is unknown a priori, the controller has to be simulated in order to determine the time—position mapping. Moreover, even if a suitable time—position mapping can be obtained, disturbances during flight will likely result in unpredictable time deviations as the proposed controllers don't know of any time annotation. For example, in the first variant the controller won't reduce the speed below the planned value following an unintended brief excess of speed. Hence, the aircraft will remain ahead of the predicted position for the rest of the flight.

In order to enable the design of a controller that is able to track the trajectory even w.r.t. time, the position has to be defined as a function over time. Using B-Splines as introduced above, this can be achieved by interpreting knots as points in time, leading to an implicit definition of velocity and acceleration:

$$\mathbf{x}(t) = \sum_{i=0}^{p+k-1} N_i^p(t)\,\mathbf{P}_i \tag{10}$$

$$\mathbf{v}(t) = \sum_{i=0}^{p+k-1} N'^p_i(t)\,\mathbf{P}_i \tag{11}$$

$$\mathbf{a}(t) = \sum_{i=0}^{p+k-1} N''^p_i(t)\,\mathbf{P}_i \tag{12}$$

3.3 Trajectory Collision Checking

One central property of B-Splines is each segment completely lying inside the convex hull of the $p+1$ control points that are active for the particular segment. In conjunction with the knot values representing points in time this allows efficient collision checking of two trajectories as sketched in Algorithm 1:

Each segment i of trajectory A is valid between τ_i^A and τ_{i+1}^A for $p \leq i < p+k$. Therefore, the algorithm first determines j_{low} and j_{high} so that the corresponding segments of trajectory B are valid at $\tau_i^A - t_{guard}$ and $\tau_{i+1}^A + t_{guard}$, respectively. t_{guard} denotes the separation in time between two UAVs that is enforced by the UTM. For each pair of i and $j \in [j_{low}, j_{high}]$ the convex hulls of the according segments are then examined for intersection. CREATEBOUNDINGBOX is expected to construct the minimal bounding box containing all given control points, expanding the edges by a certain length to the outside allowing for the required spatial separation of UAVs.

It is only if two bounding boxes intersect that the comparatively expensive function ENSURESEPARATION gets called. This function still has to discretize the corresponding segments and perform pairwise distance calculations as is the case for waypoint based approaches. However, using B-Splines this expensive function has to be evaluated for few segments only, as most trajectory pairs can be expected to have little areas of intersection, if any.

Algorithm 1 Collision Checking of Trajectories A and B

procedure COLLISIONCHECK(T^A, P^A, T^B, P^B) ▷ Returns true if trajectories collide
  $k^A \leftarrow$ length of $T^A - 2p - 1$
  $k^B \leftarrow$ length of $T^B - 2p - 1$
  $j_{low} \leftarrow p$
  $j_{high} \leftarrow p$
  for $i \leftarrow p$ to $(k^A + p)$ **do**
    while $\tau^B_{j_{low}+1} \leq (\tau^A_i - t_{guard})$ **do** ▷ Find j_{low} so $\tau^B_{j_{low}} \leq \tau^A_i - t_{guard} < \tau^B_{j_{low}+1}$
      $j_{low} \leftarrow j_{low} + 1$
      if $j_{low} \geq (k^B + p)$ **then** ▷ Beyond end of trajectory b
        return false
    while $\tau^B_{j_{high}+1} \leq (\tau^A_{i+1} + t_{guard})$ and $j_{high} < (k^B + p)$ **do** ▷ Find proper j_{high}
      $j_{high} \leftarrow j_{high} + 1$
    $Box^A \leftarrow$ CREATEBOUNDINGBOX($\mathbf{P}^A_{i-p}, \ldots, \mathbf{P}^A_i$)
    for $j \leftarrow j_{low}$ to j_{high} **do**
      $Box^B \leftarrow$ CREATEBOUNDINGBOX($\mathbf{P}^B_{j-p}, \ldots, \mathbf{P}^B_j$)
      if CHECKINTERSECTION(Box^A, Box^B) **then**
        if not ENSURESEPARATION(T^A, P^A, T^B, P^B, i, j) **then**
          return true

4 Flight Trajectory Planning Algorithm

In the last section we have shown how time annotated B-Splines can be used to precisely describe an aircraft's trajectory. Additionally we have pointed out an efficient approach to check for collisions between two given trajectories. This simplicity comes at a cost though: In general, determining appropriate knots and control points for a given geometric layout is a complex optimization problem [10, 11]. However, as has been pointed out in Section 1 the expected need for in-flight trajectory replanning requires a deterministic approach.

From (1) and (2) it can be seen that $k+1$ distinct knots and $k+p$ control points are required for a spline with k segments. As stated above, each segment $p \leq i < p+k$ is valid for $\tau_i \leq t < \tau_{i+1}$ and depends on $p+1$ control points $\{\mathbf{P}_{i-p}, \ldots, \mathbf{P}_i\}$ only.

Therefore, it seems reasonable to follow an iterative approach in which τ_{i+1} and $\mathbf{P}_i$ are determined based on τ_i and $\{\mathbf{P}_{i-p}, \ldots, \mathbf{P}_{i-1}\}$ only. For the first segment, $i = p$, sufficient values for $\{\tau_0 = \ldots = \tau_p\}$ and $\{\mathbf{P}_0, \ldots, \mathbf{P}_{p-1}\}$ have to be determined, of course. The knots are obviously defined as the point in time at which the trajectory is supposed to start. Using the initial conditions for position, velocity and acceleration, according control points can be calculated as evaluating a B-Spline at $t = \tau_0$ yields the following correlations:

$$\mathbf{x}(\tau_0) = \mathbf{P}_0 \tag{13}$$

$$\mathbf{v}(\tau_0) = \frac{3}{\tau_{p+1} - \tau_p} (\mathbf{P}_1 - \mathbf{P}_0) \tag{14}$$

$$\Leftrightarrow \mathbf{P}_1 = \mathbf{x}(\tau_0) + \frac{\tau_{p+1} - \tau_p}{3} \mathbf{v}(\tau_0) \tag{15}$$

$$\mathbf{a}(\tau_0) = \frac{6}{\tau_{p+1} - \tau_p} \left(\frac{\mathbf{P}_0 - \mathbf{P}_1}{\tau_{p+1} - \tau_p} + \frac{\mathbf{P}_2 - \mathbf{P}_1}{\tau_{p+2} - \tau_p} \right) \tag{16}$$

$$\Leftrightarrow \mathbf{P}_2 = \mathbf{x}(\tau_0) + \frac{(\tau_{p+1} - \tau_p) + (\tau_{p+2} - \tau_p)}{3} \mathbf{v}(\tau_0) + \frac{(\tau_{p+1} - \tau_p)(\tau_{p+2} - \tau_p)}{6} \mathbf{a}(\tau_0) \tag{17}$$

Additionally, it can be shown that for B-Splines of degree three the derivative of acceleration, jerk $\mathbf{j}_i$, is constant for each segment, so $\mathbf{P}_i$ can be determined using

$$\mathbf{P}_i = \mathbf{P}_{i-1} + \Delta_3(\tau_i)\Delta_2(\tau_i) \left[\frac{\Delta_1(\tau_i)\mathbf{j}_i}{6} + \frac{\mathbf{P}_{i-3} - \mathbf{P}_{i-2}}{\Delta_3(\tau_{i-2})\Delta_2(\tau_{i-1})} + \frac{\mathbf{P}_{i-1} - \mathbf{P}_{i-2}}{\Delta_3(\tau_{i-1})\Delta_2(\tau_{i-1})} + \frac{\mathbf{P}_{i-1} - \mathbf{P}_{i-2}}{\Delta_3(\tau_{i-1})\Delta_2(\tau_i)} \right] \tag{18}$$
$$\text{with} \Delta_d(\tau_i) = \tau_{i+d} - \tau_i \,.$$

Hence, the problem shifts to splitting the trajectory into reasonable spline segments for which then appropriate $\mathbf{j}_i$ and $\Delta_1(\tau_i)$ have to be determined.

Adapting [2], we assume a path to be a sequence of straight line segments, turns and hovering phases. The sequence is built so that each line segment is followed by a non-straight element and vice versa. Additionally, we require the acceleration to be zero at every transition between two elements. Hence, it seems reasonable to examine these different kinds of path elements separately.

4.1 Hover Phase

A hover phase is defined as an interval during which the aircraft's ground speed and acceleration both are constantly zero. This directly implies $\mathbf{j}_{hover} = \mathbf{0}$ and obviously Δt_{hover} equals the desired hover duration.

4.2 Turn Elements

We assume turn elements to be defined by a radius and both velocity vectors at start and end. In this paper, we require all turns to begin and end with unaccelerated and identical speed, i.e.

$$\mathbf{a}_{start} = \mathbf{a}_{end} = \mathbf{0} \quad \text{and} \tag{19}$$

$$V_{start} = V_{end}\,. \tag{20}$$

Obviously, a turn cannot be realized by a single segment, as (19) would require a zero jerk. This would result in a constant acceleration of zero for the whole segment, so the velocity couldn't be transformed accordingly.

For a two-segment approach, it can be shown that assuming symmetric segments, i.e. $\Delta t_1 = \Delta t_2$ and $\mathbf{v}_{junction} \,||\, (\mathbf{v}_{start} + \mathbf{v}_{end})$, the speed varies strongly during the turn, reaching its minimum $V_{junction} = ||\mathbf{v}_{start} + \mathbf{v}_{end}||/2$ at half turn, e.g. $V_{junction} = V_{start}/\sqrt{2}$ for a turn of 90 degrees.

Therefore, a three-segment approach is used. It seems reasonable to interpret the three segments as discrete phases of bank angle incrementation, flight at constant bank angle, and bank angle decrementation. Additionally, we will assume a symmetric layout of the turn, thereby requiring

$$\Delta t_3 = \Delta t_1\,. \tag{21}$$

For the following considerations, we use the vectors as depicted in Figure 3:

$$\mathbf{v}_{i,n} = \frac{\mathbf{v}_i}{||\mathbf{v}_i||} \tag{22}$$

$$\mathbf{n} = \frac{\mathbf{v}_1 \times \mathbf{v}_4}{||\mathbf{v}_1 \times \mathbf{v}_4||} \tag{23}$$

$$\mathbf{r}_i = \mathbf{n} \times \mathbf{v}_{i,n} \tag{24}$$

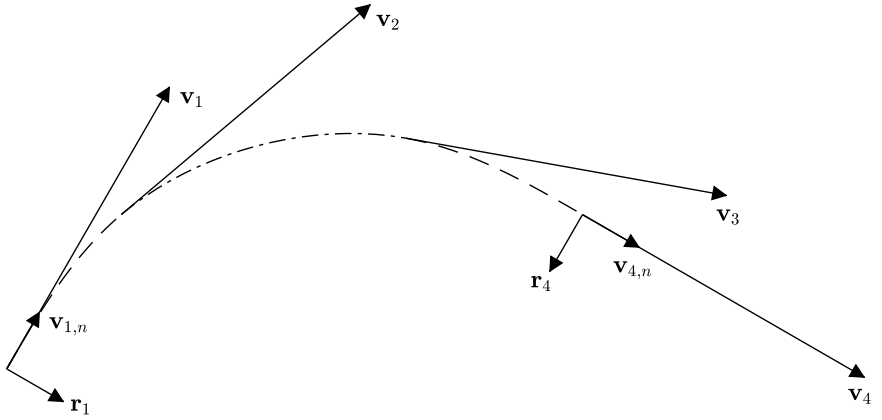

Fig. 3 Relevant Vectors for Turn Calculations

Here, $\mathbf{v}_{i,n}$ denotes the normalized velocity vector with i being 1 to 3 at the beginning of the first to third segment, respectively, and 4 at the end of the last segment. Therefore, $\mathbf{v}_1 = \mathbf{v}_{start}$ and $\mathbf{v}_4 = \mathbf{v}_{end}$. $\mathbf{n}$ is the normal vector of the plane spanned by

$\mathbf{v}_1$ and $\mathbf{v}_4$. This vector is used to calculate the normalized vector $\mathbf{r}_i$ that is perpendicular to $\mathbf{v}_i$ inside the plane and points to the inside of the turn. We can then write the jerk vectors using $\mathbf{v}_{i,n}$ and $\mathbf{r}_i$:

$$\mathbf{j}_1 = j_{1,t}\,\mathbf{v}_{1,n} + j_{1,r}\,\mathbf{r}_1 \tag{25}$$
$$\mathbf{j}_3 = j_{3,t}\,\mathbf{v}_{4,n} + j_{3,r}\,\mathbf{r}_4 \tag{26}$$
$$\tag{27}$$

With these definitions, considering (19) and (21), the velocity and acceleration vectors at start and end of the middle segment are

$$\mathbf{v}_2 = \left(V_1 + j_{1,t}\frac{\Delta t_1^2}{2}\right)\mathbf{v}_{1,n} + j_{1,r}\frac{\Delta t_1^2}{2}\mathbf{r}_1 \tag{28}$$
$$\mathbf{v}_3 = \left(V_4 + j_{3,t}\frac{\Delta t_1^2}{2}\right)\mathbf{v}_{4,n} + j_{3,r}\frac{\Delta t_1^2}{2}\mathbf{r}_4 \tag{29}$$
$$\mathbf{a}_2 = \left(j_{1,t}\mathbf{v}_{1,n} + j_{1,r}\mathbf{r}_1\right)\Delta t_1 \tag{30}$$
$$\mathbf{a}_3 = -\left(j_{3,t}\mathbf{v}_{4,n} + j_{3,r}\mathbf{r}_4\right)\Delta t_1 \tag{31}$$

Due to the posed symmetry requirement, we expect $V_2 = V_3$ as well as $A_2 = A_3$. The previous equations combined with (20) lead to

$$j_{1,t} = j_{3,t} \quad \text{and} \tag{32}$$
$$|j_{1,r}| = |j_{3,r}| \tag{33}$$

As we expect the acceleration vectors to point to the inside of the turn, i.e. in direction of r_i, from (30), (31) and (33)

$$j_{1,r} > 0 \quad \wedge \quad j_{3,r} < 0 \tag{34}$$
$$\Rightarrow j_{3,r} = -j_{1,r} \tag{35}$$

can be concluded.

For the second segment, sufficient jerk and duration have to be calculated in order to transfer $\mathbf{v}_2$ and $\mathbf{a}_2$ to $\mathbf{v}_3$ and $\mathbf{a}_3$, respectively:

$$\mathbf{j}_2 = \frac{\mathbf{a}_3 - \mathbf{a}_2}{\Delta t_2} \tag{36}$$
$$\mathbf{v}_3 = \mathbf{v}_2 + \mathbf{a}_2\Delta t_2 + \frac{\mathbf{j}_2}{2}\Delta t_2^2 \tag{37}$$
$$\Leftrightarrow \Delta t_2 = \frac{2V_1\left(\mathbf{v}_{4,n} - \mathbf{v}_{1,n}\right)}{\left[j_{1,r}\left(\mathbf{r}_1 + \mathbf{r}_4\right) - j_{1,t}\left(\mathbf{v}_{4,n} - \mathbf{v}_{1,n}\right)\right]\Delta t_1} - \Delta t_1 \tag{38}$$

This leaves Δt_1 and $\mathbf{j}_1$ being the only unknown variables. Recalling the interpretation of the turn's first spline segment as phase of building up bank angle, Δt_1 and $\mathbf{j}_1$ can be determined based on the maximum allowed lateral acceleration for this turn, $a_{r,max}$, and its rate of change, $j_{r,max}$. Assuming $j_{1,r}$ to be sufficiently perpendicular

to the velocity vector during the entire segment,

$$j_{1,r} = j_{r,max} \quad \text{and} \tag{39}$$

$$\Delta t_1 = \frac{a_{r,max}}{j_{1,r}} \tag{40}$$

can be concluded. As we want the speed to remain as constant as possible during the entire turn, by requiring $V_2 \stackrel{!}{=} V_1$ (28) can be transformed to

$$4 V_1 j_{1,t} + \left(j_{1,t}^2 + j_{1,r}^2\right) \Delta t_1^2 \stackrel{!}{=} 0 \tag{41}$$

$$\Rightarrow j_{1,t} = -\frac{2}{\Delta t_1^2} V_1 + \sqrt{\frac{4}{\Delta t_1^4} V_1^2 - j_{1,r}^2} \quad . \tag{42}$$

Examining the above equations one might note the absence of positional constraints. However, since a turn cannot resemble a perfect circular segment due to the requirement of steady transitions from and to the adjacent line segments, forcing exact positions for a given turn geometry a priori seems unreasonable. Instead, we propose to delay the exact determination of start and end point of the turn until after the calculation of the turn's jerks and intervals. Therefore, a turn is constructed with the calculated values starting from the intersection of incoming and outgoing line segments, see Figure 4. As the endpoint won't reside on the outgoing line segment, the whole turn is then moved along the first line segment.

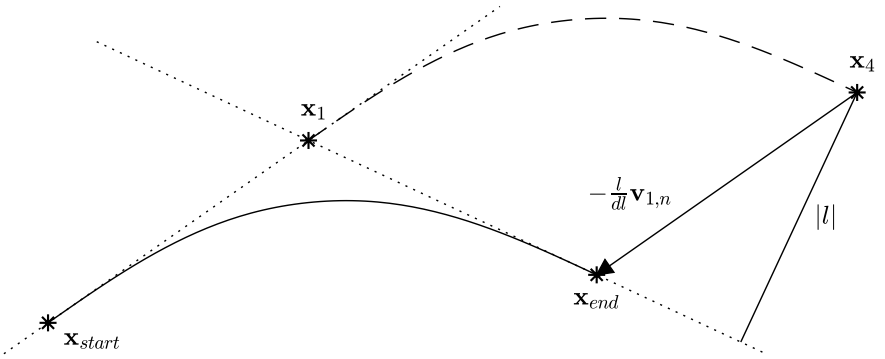

Fig. 4 Determining Start and End Position of Turn Element

Starting with $\mathbf{v}_1$, $\mathbf{x}_1$ gets propagated according to the calculated jerk—interval sequence, eventually ending in $\mathbf{x}_4$. The distance of $\mathbf{x}_4$ to the second line can be calculated as $l = (\mathbf{x}_4 - \mathbf{x}_1) \cdot \mathbf{r}_4$. Also, moving any point one unit along $\mathbf{v}_{1,n}$ increases its distance to the second line by $dl = \mathbf{v}_{1,n} \cdot \mathbf{r}_4$. Therefore, correct starting and ending points can be obtained by

$$\mathbf{x}_{start} = \mathbf{x}_1 - \mathbf{v}_{1,n} \frac{(\mathbf{x}_4 - \mathbf{x}_1) \cdot \mathbf{r}_4}{\mathbf{v}_{1,n} \cdot \mathbf{r}_4} \tag{43}$$

$$\mathbf{x}_{end} = \mathbf{x}_4 - \mathbf{v}_{1,n} \frac{(\mathbf{x}_4 - \mathbf{x}_1) \cdot \mathbf{r}_4}{\mathbf{v}_{1,n} \cdot \mathbf{r}_4} \,. \tag{44}$$

The previous equations can be used to create turn elements with nearly constant ground speed and yaw rate. In presence of wind, though, airspeed, yaw rate and commanded bank angle vary significantly. However, in many cases keeping airspeed and bank angle constant is supposed to be more important than the exact shape of a turn. As the calculations of the turn segments' intervals and jerks depend on velocity vectors and the maximum allowed lateral acceleration only, turns with constant airspeed can be calculated easily:

1. Determine the desired maximum lateral acceleration from the maximum allowed bank angle: $a_{r,max} = \tan(\Phi_{max})g$.
2. Calculate starting and ending velocities expressed both with reference to air and earth, $\mathbf{v}_{a,1}$ and $\mathbf{v}_{a,4}$ as well as $\mathbf{v}_{k,1}$ and $\mathbf{v}_{k,4}$: Using desired airspeed V_a, estimated wind velocity $\mathbf{v}_w$ and normalized direction vectors of the path at start and end of the turn, $\mathbf{d}_1$ and $\mathbf{d}_4$, Figure 5 yields

$$v_{w,i,t} = \mathbf{v}_w \cdot \mathbf{d}_i \tag{45}$$

$$|v_{a,i,r}| = |v_{w,i,r}| = \sqrt{V_w^2 - (\mathbf{v}_w \cdot \mathbf{d}_i)^2} \tag{46}$$

$$v_{a,i,t} = \sqrt{V_a^2 - |v_{a,i,r}|^2} = \sqrt{V_a^2 - V_w^2 + (\mathbf{v}_w \cdot \mathbf{d}_i)^2} \tag{47}$$

$$\mathbf{v}_{k,i} = (v_{a,i,t} + v_{w,i,t})\,\mathbf{d}_i \tag{48}$$

$$\mathbf{v}_{a,i} = \mathbf{v}_{k,i} - \mathbf{v}_w \,. \tag{49}$$

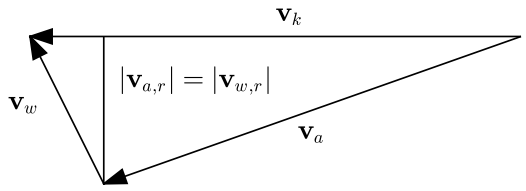

Fig. 5 Velocity Vectors Under Wind

3. Evaluate (21) to (42) with $\mathbf{v}_{a,1}$, $\mathbf{v}_{a,4}$ and $a_{r,max}$ to receive jerks and intervals.
4. Determine starting and ending point of the turn according to (43) and (44) using $\mathbf{v}_{k,1}$ and $\mathbf{v}_{k,4}$.

4.3 Straight Line Segments

Because the exact start position of a turn element can only be determined after the calculation of the turn's spline segments it seems reasonable to postpone the calculation of line segment parameters until after the processing of the succeeding turn element. Using velocity and position of the end of the previous turn as well as the start of the following turn, the geometric constraints of a straight line segment are completely given. Using the previously discussed algorithm to calculate turns it is guaranteed that both velocities are parallel to each other as well as to the line connecting both points.

Analog to the turn algorithm, a straight line segment can be split into separate spline segments of constant jerk each: In order to transform the velocity from V_{start} to V_{end} an acceleration first has to be built up from 0 to $\pm A_{max}$. For the following, we assume the maximum applicable jerk and acceleration both being constant over the whole speed range and equal in positive and negative direction. After a segment of constant acceleration, it has to be decreased to $A = 0$ again. We assume the straight line segment not to be shorter than the distance needed for this speed change. If the straight line segment is even longer, an additional spline segment with constant speed is inserted. In order to minimize the time needed for the straight line segment, this constant speed spline segment is the last segment for $V_{start} < V_{end}$ and the first segment otherwise. As Figure 6 shows, the second case equals the reflection of the first, therefore the first case is examined:

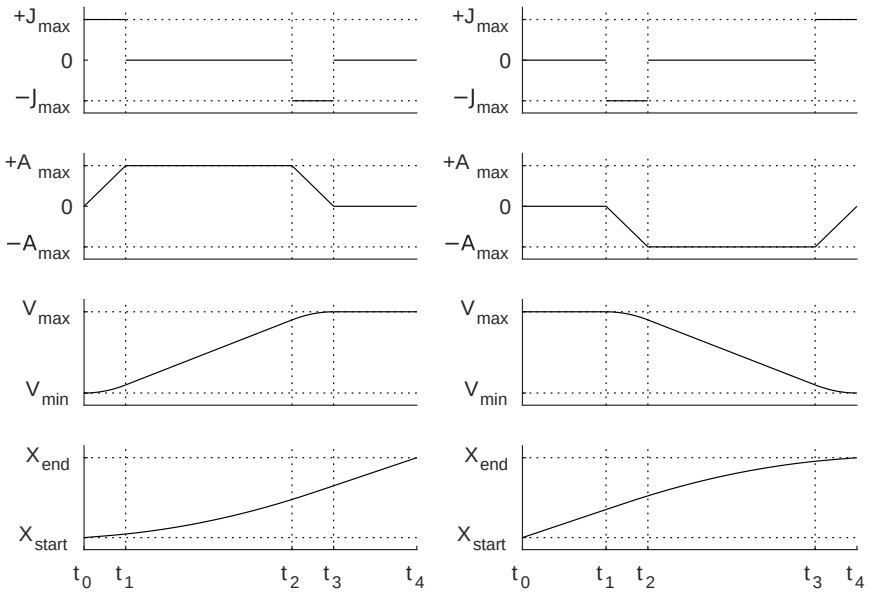

Fig. 6 From Top to Bottom: Jerk, Acceleration, Speed and Position For Straight Line Spline Segments. Straight Line Segment With $V_{start} < V_{end}$ on the Left, $V_{start} > V_{end}$ on the Right.

$$\Delta t_1 = \Delta t_3 = \frac{A_{max}}{J_{max}} \tag{50}$$

$$\Delta t_2 = \frac{V_{max} - V_{min}}{A_{max}} - \Delta t_1 \tag{51}$$

For small speed differences, $V_{max} - V_{min} < \Delta t_1 A_{max}$, Δt_2 would become negative. To prevent this, forcing Δt_2 to be zero for this case leads to

$$\Delta t_1 = \Delta t_3 = \min\left(\sqrt{\frac{V_{max} - V_{min}}{J_{max}}}, \frac{A_{max}}{J_{max}}\right). \tag{52}$$

The distance covered during this time can be calculated as

$$\Delta X = V_{min}\left(2\Delta t_1 + \Delta t_2\right) + J_{max}\Delta t_1\left(\Delta t_1^2 + \frac{3}{2}\Delta t_1 \Delta t_2 + \frac{1}{2}\Delta t_2^2\right), \tag{53}$$

therefore the duration of the additional constant speed segment with V_{max} has to be

$$\Delta t_4 = \frac{X_{end} - X_{start} - \Delta X}{V_{max}}. \tag{54}$$

4.4 Sample Trajectory

To prove the applicability of the previously presented algorithms, a sample trajectory has been created which is displayed in Figure 7 and Figure 8.

The trajectory begins in the lower left corner with a hover phase lasting 3 seconds. In order to compensate for a wind of 5 m/s blowing in negative x direction, the aerodynamic velocity has to be 5 m/s in positive x direction. Following, the aircraft accelerates in positive x direction up to an airspeed of 25 m/s. During the succeeding turn the airspeed is held at 25 m/s while the ground speed increases from 20 m/s to 23 m/s due to the wind changing from headwind to crosswind. After a subsequent straight line segment with constant speed another turn follows, this time designed for constant ground speed. Therefore, the airspeed decreases to approximately 20.3 m/s as the wind's effect changes towards tailwind.

5 Conclusion

In this paper we demonstrated how time annotated non-uniformal B-Splines can be used to precisely represent the entire trajectory of a UAV. Forming a function of position over time, this model allows calculating a UAV's position precisely for any given point in time, thereby enabling UTMs smaller separation limits.

To understand the dynamic constraints that have to be fulfilled in order to accurately describe a trajectory, we first analyzed relevant UAV flight mechanics. In

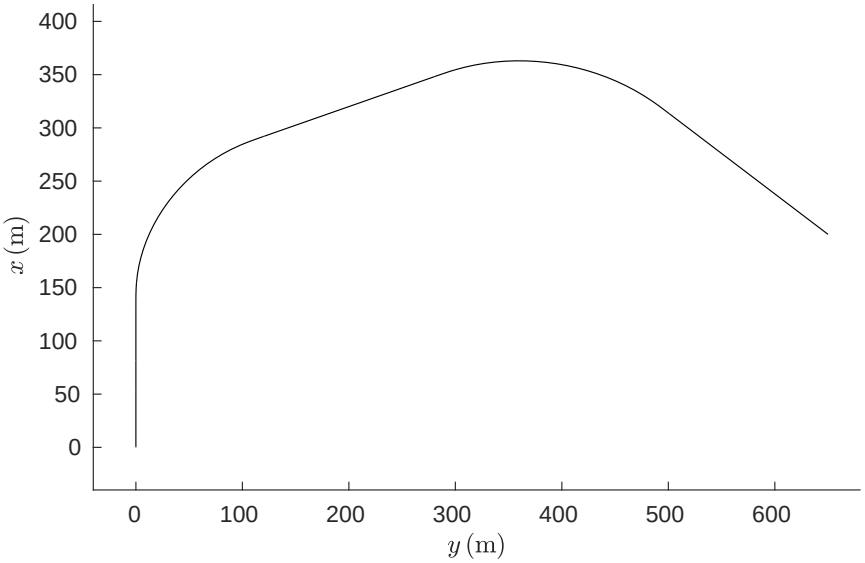

Fig. 7 Sample Trajectory Calculated With the Presented Algorithms

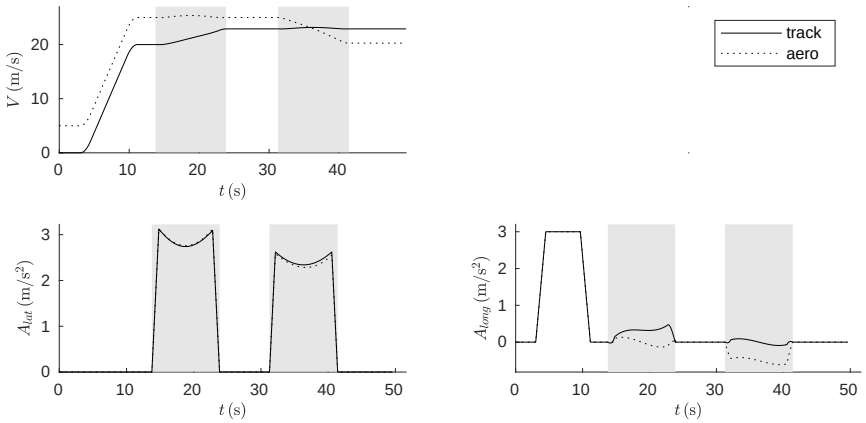

Fig. 8 Speed, Lateral and Longitudinal Acceleration for Sample Trajectory. Both With Reference to Ground and Wind. Turn Segments Shaded.

the following section, B-Splines have been expanded by interpreting their knot values as points in time, thereby transforming the B-Spline to a function over time. This transformation enables direct calculation of velocity and acceleration for any point in time. Following, we showed how the convex hull property of time annotated B-Splines allows an efficient trajectory collision detection algorithm.

As B-Splines are difficult to calculate for a given set of geometric constraints, the main part of this paper concentrated on finding appropriate B-Spline parame-

ters for an input path. In order to guarantee a quick reaction to changes, e.g. due to the detection of an obstacle, a deterministic approach had to be used. Assuming the input path to consist of a sequence of hover phases, turns, and straight line segments connecting the former two, equations were presented to determine appropriate interval—jerk sequences for these different path elements.

The presented algorithms were finally validated by analyzing a created sample trajectory. Due to the iterative structure of the proposed method, the computation time required to generate a trajectory for a given input path is extremely low: Using a Raspberry Pi Zero W which features a 1 GHz single-core ARM CPU with 512 MB RAM, the calculation of a trajectory for a path of 1201 straight line segments connected by 800 turns and 400 hover points resulting in 9594 B-spline segments takes less than 60 ms.

References

1. Balakrishnan, K., Polastre, J., Mooberry, J., Golding, R., Sachs, P.: Blueprint for the sky. Airbus (2018). URL http://utmblueprint.com
2. Barz, I., Hartmann, P., Moormann, D.: Near-term flight path adaption for tilt-wing aircraft obstacle avoidance. In: EuroGNC. Milan, Italy (2019)
3. Butt, M., Munawar, K., Bhatti, U.I., Iqbal, S., Al-Saggaf, U.M., Ochieng, W.: 4d trajectory generation for guidance module of a uav for a gate-to-gate flight in presence of turbulence. In: Int J Adv Robot Syst (2016)
4. Cichella, V., Choe, R., Mehdi, S., Xargay, E., Hovakimyan, N., Dobrokhodov, V., Kaminer, I., Pascoal, A., Aguiar, A.P.: Safe coordinated maneuvering of teams of multirotor unmanned aerial vehicles: A cooperative control framework for multivehicle, time-critical missions. IEEE Control Systems **36**, 59–82 (2016)
5. Hoffmann, G.M., Waslander, S.L., Tomlin, C.J.: Quadrotor helicopter trajectory tracking control. In: In Proc. AIAA Guidance, Navigation, and Control Conf. (2008)
6. JARUS: Jarus guidelines on specific operations risk assessment (sora) (2017)
7. Joubert, N., Roberts, M., Truong, A., Berthouzoz, F., Hanrahan, P.: An interactive tool for designing quadrotor camera shots. ACM Trans. Graph. **34**(6), 238:1–238:11 (2015)
8. Jung, D., Tsiotras, P.: On-line path generation for small unmanned aerial vehicles using b-spline path templates. In: AIAA Guidance, Navigation and Control Conference. Honolulu, HI (2008). AIAA Paper, pp. 2008–7135 (2008)
9. Lorenz, S., Adolf, F.M.: A decoupled approach for trajectory generation for an unmanned rotorcraft. In: F. Holzapfel, S. Theil (eds.) Advances in Aerospace Guidance, Navigation and Control, pp. 3–14. Springer, Berlin Heidelberg (2011)
10. Piegl, L., Tiller, W.: The NURBS Book. Springer (1995)
11. Schumaker, L.: Spline Functions: Basic Theory. John Wiley & Sons (1981)
12. Schütt, M.: Experimenteller entwurf einer basis-regelung für ein kleinstflugzeug in tiltwing-konfiguration. In: DLRK. Rostock, Germany (2015)
13. SESAR Joint Undertaking: U-space blueprint (2017). DOI 10.2829/335092