Unmanned Systems, Vol. 1, No. 1 (2013) 59–74

DOI: 10.1142/S2301385013500040

UNMANNED SYSTEMS
http://www.worldscientific.com/worldscinet/us

A Computationally Efficient Approach to Trajectory Management for Coordinated Aerial Surveillance

James Keller*,‡, Dinesh Thakur*,§, Vladimir Dobrokhodov†,¶, Kevin Jones†,‖, Mihail Pivtoraiko*,**, Jean Gallier*,††, Isaac Kaminer†,‡‡, Vijay Kumar*,§§

*GRASP Laboratory, University of Pennsylvania, Philadelphia, PA

†Department of Mechanical and Aerospace Engineering, Naval Postgraduate School, Monterey, CA

Time optimal path planning and trajectory management algorithms for air vehicles with limited on-board computing resources require an efficient approach to satisfy flight dynamic constraints needed to guarantee paths are feasible. B-spline curves enable compact definition of feasible airplane trajectories that are suited for on-board real-time computation. The design of a trajectory definition and management algorithm suited for a multi-agent persistent surveillance application is described. The proposed solution post-processes the output of a point-by-point path planner and converts it into a minimal representation. Key design requirements include minimization of mission execution time, ability to seamlessly redirect agents based on information acquired by sensor feedback, and robust adherence to mission and vehicle motion constraints. A simple coordinated aerial surveillance scenario is described and demonstrated using the algorithms presented.

Keywords: Time-optimal path planning; spline-based trajectory generation; graph search path planning; applied B-spline algorithms; coordinated aerial search.

US

1. Introduction

The task of path planning for coordinated multi-agent aerial surveillance has received much attention as the number and types of unmanned aerial vehicles (UAV) have blossomed. Unfortunately, finding optimal trajectories in a specific aerial surveillance mission for UAVs embeds a computationally intractable level of complexity[a] into such mission planning. To address this dilemma, researchers have tailored their underlying assumptions to permit the derivation of viable solutions within particular problem structures. A further concern is that once a feasible path has been determined, it is advantageous to encode it in a compact form that facilitates low bandwidth communication with on-board processors and efficient computation to enable precision navigation with minimal overhead. To address these concerns, we apply a capacitated coordinated routing solver at the ground station and then transcribe its solution into compact representations using parametric spline curves for use on-board the vehicles. B-spline curves provide a parametric framework within which trajectories can be rapidly transcribed to meet a desired path precision. Moreover, they lend themselves to rapid reassignment of agents which may be mandated as their primary surveillance sensors acquire information. Much of the foundation required for such algorithms can be found in the seminal contributions of Schoenberg[1] and de Boor[2] and many others. Their work provides a rich framework with which to define space curves and surfaces but naive application of these techniques can lead to numerically substandard performance, unnecessarily dense computational solutions, or both. In contrast, we define an approach that can accomodate real-time definition of feasible aerial trajectories. The approach also affords a high degree of flexibility with respect to task reassignment.

Received 4 February 2013; Revised 5 March 2013; Accepted 31 March 2013; Published 10 June 2013. This paper was recommended for publication in revised form by the Editors-in-Chief.
Email Addresses: ‡*jfkeller@seas.upenn.edu*, §*tdinesh@seas.upenn.edu*, ¶*vndobrok@nps.edu*, ‖*kdjones@nps.edu*, **mihailp@seas.upenn.edu*, ††*jean@cis.upenn.edu*, ‡‡*kaminer@nps.edu*, §§*kumar@seas.upenn.edu*

[a]This problem can be mapped to the Traveling Salesman or Team Orienteering Problems, both of which are recognized to be NP-hard.

This article starts with defining the scenario to which our solutions are applied, outlining requirements and then describing the routing solver. We then describe how its solutions are managed to maximize precision and efficiency of mission execution. Finally, the hardware and software systems used to demonstrate these concepts are described before conclusions are provided. Key foundational concepts are presented in the appendix for reference.

2. Mission Scenario

The envisioned scenario considers a coordinated multi-UAV search for a set of noncooperative ground targets from a prioritized list. The list is built based on the preliminary intelligence that associates a set of probable locations for each target that should lead to their discovery as the mission unfolds. Locations could be specific geo-referenced coordinates or traverses through specified sections of road or terrain. The team of UAVs is tasked to work its way through the target list. A capacitated routing planner develops a flight plan for each vehicle for the highest priority target. The top level planner partitions the complexity of each vehicle plan into a series of flight segments with each one being terminated in a sub-goal point drawn from the set of probable locations. These plans are specified as a set of geo-referenced points to satisfy vehicle motion and mission oriented constraints. From these sets the proposed algorithm develops spline-based path definitions for the vehicle autopilot subsystems. Once a target is discovered and confirmed, the team is routed to search for the successive priority. Discovery could occur at any time but is most probable when the vehicles observe the sub-goal locations. Consequently, after finding spline-based plans to match the original specifications, each plan is further subdivided until splines for all vehicles between sub-goals have equal duration based on the time that one of the vehicles will arrive at a sub-goal. This subdivision process is illustrated in Fig. 1. An exception to this subdivision process is made if any segment is shorter than 10 s in duration.[b] In this case, the subdivision process is deferred at the sub-goal that generated the short segment until the one that follows. Since the original plans are only roughly equivalent in duration, towards the end of the search for a specific target the full team may no longer be required, as depicted on Fig. 1 where two vehicles are used. If the mission reaches this stage, the vehicles that have completed their routes are available for the routing planner to be called for the next target on the list and can be tasked for the next

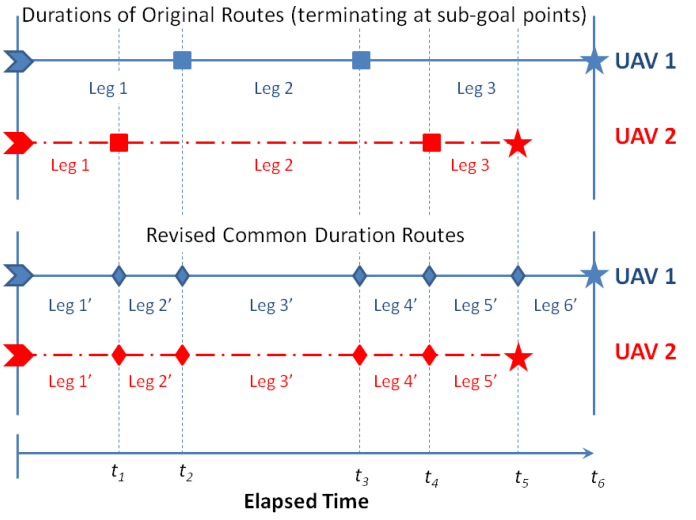

Fig. 1. (Color online) Example scenario for two UAVs: paths from route planner (top panel), sub-divided paths of equal duration (bottom panel).

search directly from their current locations, while the remaining ones are tasked from the terminal points of their still unfolding plans (Leg 6 in the example on Fig. 1, bottom panel). Alternatively, if an object of interest is detected at any point, that vehicle can be directed to an orbiting flight plan to confirm the identity of the target while other vehicles are rerouted from a future point on their current plan based on the time required for the solver to replan from the moment positive identification is made. In all cases, it is advantageous to keep the vehicle plans closely coordinated. The examples presented are derived for constant speed. While the autopilot controller can vary speed as desired, this degree of freedom is reserved for disturbance rejection to ensure high fidelity to constant speed paths. The justification for the constant speed assumption will be provided in Sec. 3, which follows.

3. Requirements for Aerial Vehicle Path Planning Algorithms

In order to derive feasible paths for a fixed-wing UAV, a planner needs to take into account the flight dynamics response of the vehicle and its controls because these often introduce significant control authority limits and response lags which cannot be adequately captured in purely kinematic models. A simple, physically representative, dynamic model of a fixed-wing UAV suitable for embedding in a path planner can be derived from the following assumptions:

- The vehicle has automatic turn coordination. In other words, we assume vehicle longitudinal axis remains aligned with the velocity vector for consistency with most autopilots.

[b] A minimum segment duration of 10 s is imposed for the practical considerations that the planner may require a duration on this order to converge and the coordination control law embedded in the autopilot also may require a comparable time to eliminate the coordination errors at the start of a segment.

- Turning flight is accomplished by rolling the vehicle about its longitudinal axis to a bank angle to generate a centripetal acceleration. Moreover, the magnitude of bank angle is bounded: $\phi < \phi_{\max}$, since banked flight is typically constrained by vehicle aerodynamic performance and autopilot operational limits, especially for small vehicles.
- The rate of roll response is constrained by an effective roll damping factor, which is typically denoted as $\frac{\partial L}{\partial p}$ (where L represents the rolling moment of the vehicle, normalized by inertia, which is induced by its roll rate, denoted by p). The magnitude of this parameter is set by the vehicle configuration. A typical time constant for this response for small surveillance UAVs would be on the order of 1 to 4 s.
- The magnitude of roll rate is bounded: $p < p_{\max}$, since roll rate is constrained by the control authority of vehicle and its response dynamics.
- There are minimum and maximum bounds on airspeed since it is constrained by flight dynamics and the magnitude of installed power.

Model complexity can be greatly reduced by removing the vertical degree of freedom and working with constant altitude planar paths and further reduced by fixing airspeed to be constant. The fixed altitude constraint is compatible with the mission context because the height above ground level is likely set by on-board sensor field of view or resolution. The fixed airspeed constraint is made practical by consideration that surveillance UAVs tend to operate between their maximum endurance speed and their maximum range speed.[c] For many configurations the difference between these can be small, especially at high density altitude. Therefore, plans are generated for constant speed at the middle of this range, so the autopilot has sufficient control authority to reject disturbances, such as winds. A consequence of fixed airspeed and altitude is that the pitch angle, θ, will be small enough to neglect in the turning equations of motion. These assumptions can be used to define a suitable set of equations of motion, such as those developed by Seckel[3] or Stengel.[4] The fundamental concept that must be captured by a planner is that turning is accomplished by rolling the vehicle for which there is an associated response lag. As a consequence, the response rate to initiate a bank angle is governed by the stability derivative $\frac{\partial L}{\partial p}$. For constant speed applications, minimum turning radius is directly set by the upper bound on bank angle ($r_{\min} = V^2/g\tan(\phi_{\max})$), where g is the gravitational constant and $\phi_{\max}$ is the maximum angle of bank that can be achieved. These physical constraints tend to reduce the fidelity of path planners which abstract dynamics away to work strictly with simplified kinematic bounds on turning radius. At present, most small UAVs exhibit response dynamics significant enough to affect the shapes of their turning paths. These assumptions can be used to define the following equations of motion for constant velocity.

$$\dot{p} = L_p p + L_{\delta_a}\delta_a. \tag{1}$$

$$\dot{\phi} \approx p. \tag{2}$$

$$\dot{\psi} = \frac{g\tan\phi}{V}. \tag{3}$$

$$\dot{x} = V\cos\psi. \tag{4}$$

$$\dot{y} = V\sin\psi, \tag{5}$$

where the definitions of the variables are provided in Table 1:

This system, which is inherently nonlinear even though aerodynamic contributions are linear, has five dynamic states. While there are typically two control inputs, as noted, we work with a constant speed command matched to mission requirements so there is one control input for the planner. These equations are:

$$[\vec{\mathbf{x}}] = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} p \\ \phi \\ \psi \\ x \\ y \end{bmatrix}. \tag{6}$$

$$\mathbf{u} = [\delta_{\text{aileron}}]. \tag{7}$$

Table 1. Nomenclature.

Variable	Description	Units
V	airspeed along flight path	m/s
p	roll rate	radian/s
δ_a	aileron displacement, scaled to represent roll rate command	radian/s
$\frac{\partial L}{\partial p}$	roll damping normalized by inertia, abbreviated as L_p	1/s
$\frac{\partial L}{\partial \delta_a}$	roll control derivative normalized by inertia, abbreviated as $L_{\delta a}$	$\frac{\text{radian}}{\text{s}^2} / \frac{\text{radian}}{\text{s}}$
ϕ	roll or bank angle, (positive for a RH rotation about the longitudinal axis)	radian
ψ	inertial heading (positive clockwise with respect to north)	radian
g	acceleration due to gravity	9.80665 m/s^2
x	x inertial position (positive for displacement north from initial condition)	m
y	y inertial position (positive displacement east from initial condition)	m

[c] Endurance and range are measured with respect to a what can be accomplished on a single load of fuel. In this context, endurance refers to time aloft, while range refers to distance traversed.

4. Time-Optimal Cooperative Path Planning

While the focus of this article is the structure and management of vehicle trajectories, we apply these concepts to a graph-based path planner as a sample case. In this work, instead of formulating route planning as an optimization problem, we map it to a graph-search. Most graph-search approaches assume the costs between locations are given and do not address the dynamic constraints of the agents. In order to determine time-optimal cooperative paths for a designated fleet of vehicles for a reconnaissance/search type mission, we start by identifying the locations of sites to be observed and mission duration. We then apply a variety of path control primitives in a graph-based planner to solve a time-critical capacitated vehicle routing problem. The motion primitives are constructed to capture dynamic constraints. For reference, the primitives for turning from level roll attitude are shown in Fig. 2. A variety of search techniques are employed to ensure that the planner will converge. Simpler algorithms are used if there are no obstacles. In the event, there are stationary obstacles, the search technique is adjusted accordingly to account for them.

Consider a task with N agents and M waypoints. Then, define the start coordinates for the ith agent to be S_i, the goal coordinate to be G_i where $1 \leq i \leq N$. We denote the jth waypoint by W_j where $1 \leq j \leq M$. If time T_{MAX_i} is allotted for each of the N agents for its travel from S_i to G_i, we need to find out how many of the additional M waypoints can be covered by each agent within their respective time, T_{MAX_i}, without any repetitions. The agents are modeled as non-holonomic vehicles with a minimum turning radii R_i in an abstraction of Eqs. (1) through (5). Our approach is based on graph-based representations. We construct two levels of graphs: a high level graph G_H and a low level graph G_L. For a single agent A_i the graph G_{H_i} consists of the S_i, G_i, W_j coordinates and the transition times between them. It is an abstraction of the problem. The graph G_{L_i} is a representation of the actual obstacle-filled environment. Motion primitives are applied at this level in the computation of actual transition times between any two vertices in G_{H_i}. We formalize the planning problem for multi-agent goal-directed path refinement with time constraints as the problem of finding a set of paths through N graphs G_{H_i}.

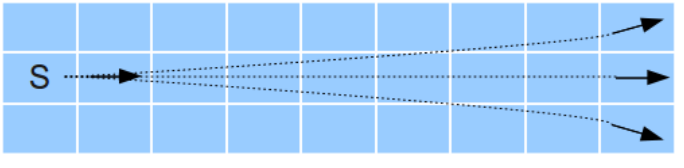

Fig. 2. (Color online) Example of motion primitives for turning flight starting from level roll attitude.

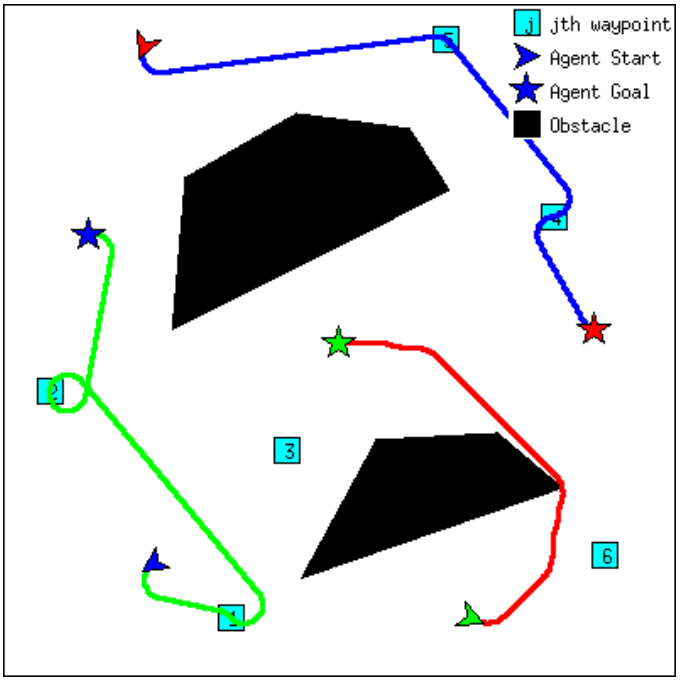

Fig. 3. (Color online) The planned paths for an environment with six waypoints, visited by three agents, while avoiding obstacles.

The constraints here are that each agent A_i should cover as many waypoints and reach the goal G_i within the time limit T_{MAX_i}. The A* search is perhaps one of the most popular methods for executing a graph search that ends the least-cost path from a given initial state to a goal state.[5] Both the graph G_L and G_H searches are done using the A* algorithm. For our scenario, which is multi-agent, we implement a greedy search which runs a series of graph searches for each agent. A sample of the planner output is presented in Fig. 3. The format of the result is fairly representative of path planners using graph-based or Voronoi decomposition to compute a solution. It is an ordered set of path points for each vehicle. The remainder of this article is focused on the subsequent management of each path with attention to computational efficiency.

5. On Time Critical Path Following and Coordination

To provide coordinated path following of multiple UAVs as, for example, the ones described in Fig. 3 a new approach to path following and coordination was developed in Ref. 6. This approach uses UAV attitude to follow a given path while using its speed to follow a desired speed profile and to coordinate its progress along this path with other UAVs.

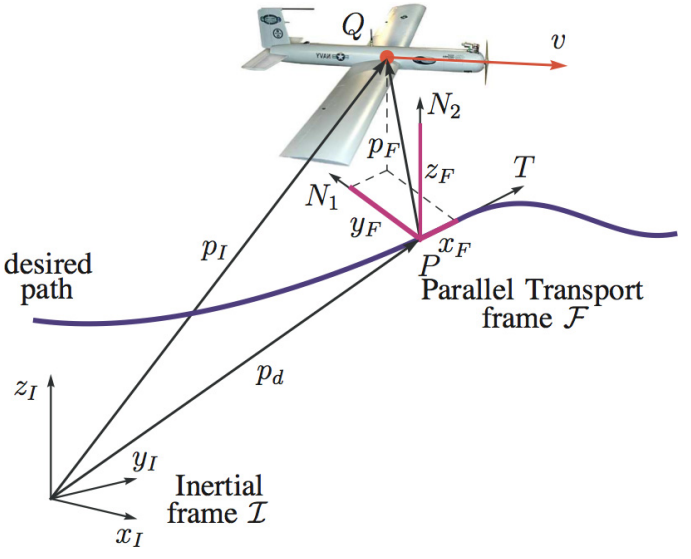

Fig. 4. (Color online) Geometry of the path following problem.

The geometry of the path following problem of single UAV is presented next in Fig. 4. The key idea of the path following control algorithm is to use the vehicle's attitude control effectors to follow a *virtual target vehicle — P* running along the desired path p_d defined in the inertial space $\mathcal{I}$. The virtual target vehicle is the degree of freedom that enables a single operator to define the pace of progression of the multiple UAVs mission. For the purpose of defining the relation between the desired p_d and actual state p_I of single UAV, we introduce a *parallel transport frame* $\mathcal{F}$[7] attached to this virtual target P: the unit vector $T(\ell)$ defines the tangent direction to the path at the point determined by relative position on the path ℓ, while $N_1(\ell)$ and $N_2(\ell)$ define the normal plane perpendicular to $T(\ell)$. Unlike the Frenet–Serret frame, parallel transport frames are well defined even when the path has a vanishing second derivative (straight line segment). Here $\ell \in [0, \ell_f]$ is a free length variable that defines the position of the virtual target vehicle along the single UAV path. Then a generalized error vector p_F between this moving coordinate system $\mathcal{F}$ and a frame attached to the center of gravity Q of actual vehicle can be easily defined as $p_F = p_I - p_d$. Using this geometrical interpretation of the path following task formulated around given a spatially defined feasible path p_d, the control problem is reduced to driving this generalized error vector p_F to zero by using vehicle's attitude control effectors only, while following an arbitrary speed profile.

The previously reported result[6] addresses precisely this problem: *For a given UAV, design feedback control laws for pitch rate $q(t)$, yaw rate $r(t)$, and rate of progression $\dot{\ell}(t)$ of the virtual target along the path such that the path-following generalized error vector $p_F(t)$ converges to a neighborhood of the origin with a guaranteed rate of convergence, regardless of the (feasible) temporal assignments of the mission.*

The solution[6] to the path following problem for the ith vehicle is independent of the desired speed profile $v_{d,i}$ and uses only local measurements for feedback. The path following control laws $q(t)$ and $r(t)$ represent outer-loop guidance commands that are to be tracked by the vehicle to guarantee the safety and success of the cooperative mission. Note, that this solution for path following control differs from standard back-stepping approach; in the new solution the final path following control law exploits all the capabilities of the existing on-board autopilot and preserves its stabilizing and tracking performance.

In turn, to enforce the temporal constraints of the mission, the speed of the vehicles is adjusted based on coordination errors (exchanged over a time-varying network) of virtual targets. In particular, the outer-loop coordination control law is developed to provide a correction to the desired speed profile $v_{d,i}$ obtained in the trajectory generation step and to generate a total speed command $v_{c,i}$. This speed command is to be tracked by the ith vehicle to achieve coordination in time.

The coordination task is first formulated as a consensus problem, in which the objective of the fleet of vehicles is to reach an agreement on some distributed variables of interest. The coordination variable required for this purpose needs to encode the relation between the temporal assignment of the mission (for example, the desired mission duration t_d) and the parametric length of the generated path $\ell_{f,i}$ of each UAV; this length is one of the key parameters that define the "shape" of the desired trajectory.

To this end, define $\ell'_{d,i}(t_d)$ be the desired normalized ($\ell'_{d,i} \in [0,1]$) curvilinear abscissa of the ith UAV along its path at the desired mission time t_d, which is given by

$$\ell'_{d,i}(t_d) := \frac{1}{\ell_{fi}} \int_0^{t_d} v_{d,i}(\tau)\, d\tau.$$

The trajectory-generation algorithm ensures that the desired speed profiles $v_{d,i}$ satisfy flight dynamics conditions, which implies that the following bounds hold: $0 < v_{\min} \le v_{d,i} \le v_{\max}$, $i = 1, \ldots, n$, where $v_{\min}$ and $v_{\max}$ denote, respectively, minimum and maximum operating speeds of the UAVs in the mission. Considering the definition of $\ell'_{d,i}(t_d)$ and the fact that $0 < v_{d,i}$, it follows that $\ell'_{d,i}(t_d)$ is a strictly increasing continuous function of t_d mapping $[0, t_d^*]$ onto $[0,1]$. In turn, define $\eta_i : [0,1] \to [0, t_d^*]$ as the inverse function of $\ell'_{d,i}(t_d)$ that is also a strictly increasing continuous function of its argument. Then, letting $\ell'_i(t) := \ell_i(t)/\ell_{fi}$, we introduce the new time-variables

$$\xi_i(t) := \eta_i(\ell'_i(t)), \quad i = 1, \ldots, n.$$

Observe that for any two vehicles i and j, if $\xi_i(t) = \xi_j(t) = t'_d$ at a given time t, then $\ell'_i(t) = \ell'_{d,i}(t'_d)$ and $\ell'_j(t) = \ell'_{d,j}(t'_d)$,

which implies that at time t the target vehicles corresponding to UAVs i and j have the desired relative position at the desired mission time t'_d. Moreover, if $\dot{\xi}_i(t) = 1$, then at time t the ith virtual target travels at the desired speed, $\dot{\ell}_i(t) = v_{d,i}(\xi_i(t))$. Therefore, the variables $\xi_i(t)$ represent the desired measure of vehicle coordination and are referred to as *coordination states*, while the functions $\eta_i(\cdot)$ are called *coordination maps*.

Using the definition of coordination states, we define the problem of time-critical cooperative path following for a fleet of n UAVs as follows: *Given a fleet of n vehicles supported by an inter-vehicle communications network and a set of desired 3D time trajectories $p_{d,i}(t_d)$, design feedback control laws for pitch rate $q(t)$, yaw rate $r(t)$, and speed $v(t)$ for all vehicles such that for each vehicle i, $i = 1, \ldots, n$, the path-following error vector $p_{F,i}(t)$ converges to a neighborhood of the origin*; *and for each pair of vehicles i and j, $i, j = 1, \ldots, n$, the coordination errors $(\xi_i(t) - \xi_j(t))$ and $(\dot{\xi}_i(t) - 1)$ converge to a neighborhood of the origin.*

The proposed decentralized control law[6] assumes that each vehicle is allowed to exchange only its coordination parameter $\xi_i(t)$ with its neighbors. These parameters are transmitted over the possibly time-varying communication topology. The resulting coordination control law has a proportional-integral structure that provides disturbance rejection capabilities at the coordination level. Moreover, the benefit of this approach is that only information about the virtual targets is exchanged over the network, instead of exchanging information about the actual vehicles. This is in contrast to most literature in cooperative control (see for example, the survey,[8]) where authors solve cooperative problems by exclusively exchanging actual agent feedback over the network. The advantages of using "virtual" information in consensus problems was illustrated in Ref. 9.

Figure 5 shows the overall cooperative path-following control architecture for the ith vehicle. It can be observed that the control architecture exhibits a multiloop control structure in which an inner-loop controller (autopilot) stabilizes the UAV dynamics, while guidance outer-loop controllers are designed to control the vehicle kinematics, providing path-following and coordination capabilities.

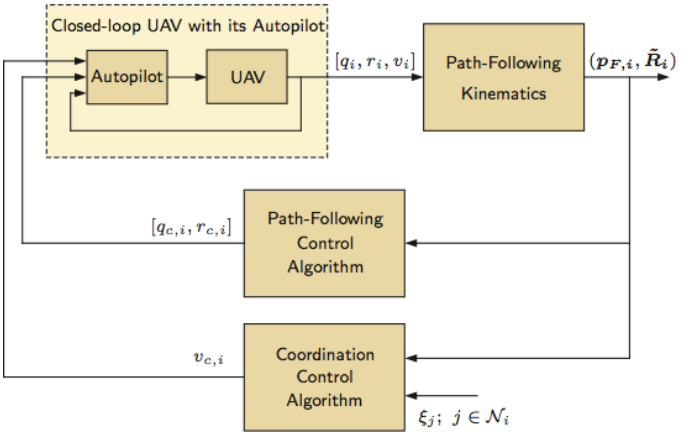

Fig. 5. (Color online) Coordinated path-following architecture for the ith vehicle.

It is proven in Ref. 10 that, if the connectivity of the communications graph verifies a specific persistence of excitation like condition and the initial conditions are within a given domain of attraction, then there exist control gains for the path-following control law and the coordination control laws that solve the time-critical cooperative path-following problem with guaranteed rates of exponential convergence, while ensuring at the same time that the speed of each UAV satisfies $v_{\min} \leq v_i(t) \leq v_{\max}$ for all $t \geq 0$. It is shown in Ref. 10 that the QoS of the network limits the achievable guaranteed rate of convergence of the coordination-error dynamics. The results illustrate that, as the communications graph becomes continuously connected pointwise in time, the convergence rate of the coordination-error dynamics can be set arbitrarily high by increasing the coordination control gains.

Finally, for more details about the development and the solution of both the path following and the coordination problems an interested reader is referred to Ref. 6.

6. Trajectory Smoothing

6.1. *Identifying the appropriate polynomial degree for aerial vehicle trajectory representation*

While minimization of the polynomial degree used to represent a trajectory is a key factor towards minimization of computational complexity, it is important to ensure that there are sufficient degrees of freedom with which to replicate feasible flight paths with precision. To ascertain an appropriate degree for the polynomial basis functions used in parametric curves, we apply the concept of differential flatness, developed by Martin *et al.*,[11] to the underlying dynamics. Starting with the assumptions developed in Sec. 3, we show that a functional representation of the position, $[x, y]$, is sufficient to capture the underlying equations of motion. For a system with two control inputs, the *flat* output vector, **z**, should contain two flat outputs if the dynamics can be shown to be flat.[12] The inertial coordinates, x and y, are the most obvious choices for planar flat outputs since they are the states that would be tracked by a controller executing a path. In the general sense, the two inputs required to define planar UAV paths are speed and turning controls, so the mathematical dimension of controls = number of flat outputs is satisfied. To minimize model complexity, we constrain the speed control to be constant. The turning control is still sufficient to reach any position in

the $x-y$ plane so we retain these as outputs.

$$\mathbf{z} = \begin{bmatrix} x \\ y \end{bmatrix}. \tag{8}$$

Using these to express the states and controls (developed earlier in Eqs. (1) through (5)) leads to the following expressions. For constant speed, V, we have:

$$V = \sqrt{\dot{x}^2 + \dot{y}^2}. \tag{9}$$

The heading of the vehicle, ψ, defines the orientation of the velocity vector. It may be directly expressed in terms of the flat outputs as:

$$\psi = \arctan(\dot{y}, \dot{x}). \tag{10}$$

The curvature of the path, κ, can be expressed in terms of the differential equations as:

$$\kappa = \frac{g\tan\phi}{V^2} = \frac{g\tan\phi}{(\dot{x}^2+\dot{y}^2)} = \frac{\dot{x}\ddot{y}-\dot{y}\ddot{x}}{(\dot{x}^2+\dot{y}^2)^{\frac{3}{2}}}. \tag{11}$$

This expression can be used to simplify the derivation of the roll attitude, ϕ, in terms of the derivatives of the flat outputs.

$$\phi = \arctan\left\{\frac{\dot{x}\ddot{y}-\dot{y}\ddot{x}}{g(\dot{x}^2+\dot{y}^2)^{\frac{1}{2}}}\right\}. \tag{12}$$

Differentiating this expression with respect to time yields the expression for roll rate in terms of the flat outputs:

$$p \approx \dot{\phi} = \left\{\frac{1}{1+\left\{\frac{\dot{x}\ddot{y}-\dot{y}\ddot{x}}{g(\dot{x}^2+\dot{y}^2)^{\frac{1}{2}}}\right\}^2}\right\} \times \left[\frac{1}{g(\dot{x}^2+\dot{y}^2)^{\frac{1}{2}}}\left\{\frac{-(\dot{x}\ddot{y}-\dot{y}\ddot{x})(\dot{x}\ddot{x}+\dot{y}\ddot{y})}{(\dot{x}^2+\dot{y}^2)} - \dddot{y}\ddot{x} + \dot{x}\dddot{y}\right\}\right]. \tag{13}$$

Equation (13) expressed in terms of the flat outputs x and y can be differentiated to yield Eq. (1), expressed in terms of the flat outputs, which can then be solved directly for the control input, δ_{aileron} in terms of x and y and their derivatives. This expression, which includes derivatives of x and y to fourth order, fully defines the control input required to generate a trajectory expressed in terms of the planar inertial coordinates x and y. These expressions ensure the equations of motion defined by (1) through (5) are satisfied. Note, the configuration constants, which constrain the physical system dynamics only appear at the fourth derivative level, where control inputs can be determined. In this regard, a curve must embed constraints at this level to be truly feasible. Since at this stage we are ultimately interested in the polynomial degree, d, required to represent this type of motion, we can conclude the minimum degree with which to express nonzero time derivatives to fourth order is $d \geq 4$. Consequently, our parametric trajectory definitions are based on basis functions of degree 4, which are defined over a range of the their parameter, τ, and have the following form:

$$\begin{aligned} \boldsymbol{r}(\tau) &= \sum_{m=0}^{M} N_{m,d}(\tau)\boldsymbol{p}_m, \\ \boldsymbol{p}_m &\equiv \text{ a set of } M+1 \text{ geometrically configured control points in } \mathbb{R}^2, \\ N_{m,d}(\tau) &\equiv \text{ a set of } M+1 \text{ basis functions of degree } d. \end{aligned} \tag{14}$$

Development of flight plans that satisfy dynamic constraints directly in a parametric format is an ongoing thrust of our research. Computationally compact transcription of curvature and rate of change of curvature constraints into this format is still an emerging element of our work. In this article, we apply the planner described in Sec. 4.

6.2. *Developing parametric representations of trajectories*

Parametric functions provide a compact framework with which to represent trajectories. However, even though it may be attractive to consider the parameter as representive of arc length distance along a path or time elapsed from a starting point, this representation is not mathematically feasible except in simple cases such as straight lines or circular curves (whose coordinates are linear or quadratic functions respectively). See Ref. 13 for details of this complicating issue. As a consequence, parametrically defined paths must account for the irregular correspondence between parameter value and arc distance along a path when tracking progress along a curve in order to be useful.

6.2.1. *B-spline trajectory definition*

While parametrically defined curves can be expressed with a diverse variety of polynomial basis functions, we apply B-splines. B-splines are a general framework for parametric splines. See Ref. 14 for a comprehensive description. Like other parametrically based methods, a set of basis functions is used in conjunction with a set of geometrically configured control points. The linear combination of control points and basis functions constitutes the curve over a specified range of an independent parameter. Planar curves can be defined in inertial space as: $\boldsymbol{r}(\tau) = [x(\tau), y(\tau)]$, where τ varies over

the range $0 \leq \tau \leq \tau_{\max}$. Unlike other basis function sets, B-spline curves are not necessarily defined over the entire domain of the parameter. They use a finite sequence of points, within the parameter space, designated as knots in order to derive and anchor the basis functions. See appendix for computational details of how the basis functions are computed as function of the knots and the parameter. The basis functions, $N_{m,d}(\tau)$ are actually a set of piecewise of polynomials that are continuous across each knot. The B-spline basis has the following features that make it particularly suited to our application. The basis is structured to provide compact support so that individual basis functions only span $d+1$ knots, $N_{i,d}(\tau) > 0$ in the open interval of knots, (τ_i, τ_{i+d+1}) and zero everywhere else. Consequently, moving control points outside of this range will have no effect on a curve. The basis functions constitute a partition of unity whereby the sum of all functions for any parameter value in the full basis span of the knot vector $= 1$. The main advantage this approach provide is that the curve will reside within the convex hull of the control points (since each function is ≥ 0). The polynomial degree of these basis functions is not driven by the number of control points, so a large amount of control points does not imply high order polynomials. Note, the polynomials are piecewise continuous functions across knot spans. Lastly, continuity across curve segments is built into the basis functions and the degree of continuity across each knot span can be selectively controlled. B-splines permit a curve to be tailored by the manipulation of the positions of the control points in inertial space, the values of the knot sequence which anchor the basis functions within the parameter space, and the degree of the polynomials used to construct the basis functions. The only requirements for a viable knot vector are:

- For $M+1$ control points and basis functions of degree d, there must be $M+d+2$ knots, since each basis function spans $d+1$ knots.
- The first knot, τ_0, must satisfy $\tau_0 \leq \tau_{\min}$, the minimum parameter over which the curve is defined.
- The knots form an increasing sequence: $\tau_i \leq \tau_{i+1}$.
- The last knot, τ_{M+d+2}, must satisfy $\tau_{\max} \leq \tau_{M+d+2}$.

When the knot sequence is strictly increasing, the functions do not span the entire space, so the curve is taken as the portion over which there is a basis for polynomials of degree d, which in our case is four. In our application, it is important to interpolate initial and final points from the set determined by our graph-search planner. To accomplish this, the initial and final knots are repeated $d+1$ times to constitute what is denoted as a clamped spline. We also uniformly space the internal knots to provide approximate uniform support in the determination of the splines and their relation to the control points. Note, B-spline basis functions are implicitly geometrically configured since they

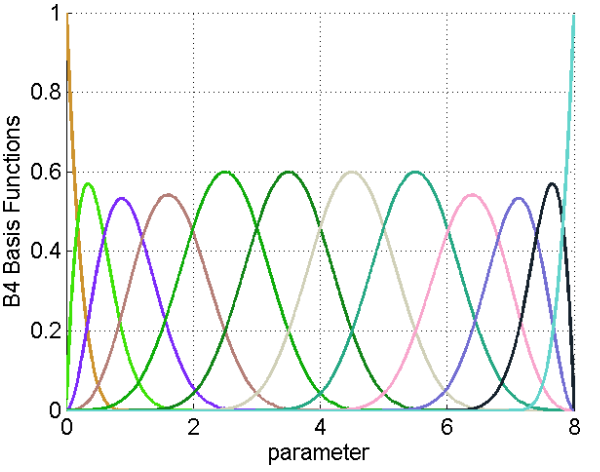

Fig. 6. (Color online) Quartic B-splines defined for *clamped* uniform knots over the range 0 to 8 (12 control points).

are related to the parameter range of the curve. By uniformly spacing the knots, the basis functions tend to be relatively equally spaced with respect to the curve parameter. While nonuniform knot spacing is supported, without *a priori* information for the shape of a curve, it is not practical to develop spline curves in this manner because it yields a set of nonuniform basis functions that can result in curve distortions. Nonuniform knot spacing is best applied to curve refinement once a desired shape has been established. Consequently, our knot sequence starts at 0 and ends at $M+1-d$. For example, a knot sequence that supports 12 control points is: $[0,0,0,0,0,1,2,3,4,5,6,7,8,8,8,8]$. The range of the parameter over which the B-splines form a basis in this case is $[0,8]$, as illustrated in Fig. 6 (for which there are $d+1$ nonzero functions across each knot span).

6.2.2. *Curve approximation using B-splines*

The output of the planner described in Sec. 4 is a lengthy sequence of points between desired target or sub-goal locations. Spline interpolation of every point in these sequences is not necessary or computationally practical. We set up a least-squares solution to locate $M+1=12$ control points to match a spline curve to the original set of points. The selection of 12 control points is a compromise between path complexity and the processing speed of the spline-based algorithms. The first step is to assign a parameter to each point in the sequence. We apply the *Chord-based* method using the Euclidean distance between $L+1$ points, since this will approximate an arc-length parameterization[15] for our curve.

$$\tau_0 = 0 \quad \tau_l = \tau_{l-1} + \|q_l - q_{l-1}\|, \quad \text{for} \quad l = 1, 2, \ldots, L. \tag{15}$$

We then scale the result so that the parameter will have the same domain as the knot sequence.

$$\text{total chord length} = \sum_{l=1}^{L} \tau_l,$$
$$\text{scale factor} = \frac{M+1-d}{\text{total chord length}}. \tag{16}$$

The least squares solution to approximate $L+1$ data points, $q_l \,|\, l = 0 \ldots L$, with $M+1$ control points, p_m, using quartic B-splines is developed as two steps. First, boundary conditions are used to directly determine the first and last control points. Next, the internal control points are found through the least squares algorithm. Computational details of this derivation may be found in Appendix C. A critical assumption is that there must be at least one data point in the span between each pair of knots: τ_i to $\tau_{i+1} \,|\, i = 0 \ldots M+d+1$. Our route finder uses a close geometric point spacing, so this is not an issue. Having determined the B-spline solution, in order for it to be directly applicable to vehicle navigation, as described in Sec. 6, we need to be able to compute the appropriate point on the curve associated with the current location of the vehicle. In this regard, it is common to compute arc-length for an appropriately dimensioned set of parameters using a quadrature algorithm and then to determine another B-spline that interpolates the parameter — arc-length pairs, $(S(\tau), \tau)$. The precision of this approach is governed by the size of the set used for this complementary spline curve and the variation of curvature in the path. Alternatively, to secure high precision for an arbitrary path, we use a three-point Gauss quadrature to determine arc-length at each knot along the curve and then apply a Newton iteration solver between knots to find the parameter associated with a specific arc-length, as described in Appendix B.

6.3. *Spline subdivision*

Once the spline curves and their durations have been determined for all vehicles and mission legs as depicted in upper panel of Fig. 1, we subdivide these into equal duration curves[d] for all vehicles until only one remains to complete the last path by itself, as depicted in the figure. For two agents, the splines for each are taken pairwise in sequence and the longer of the spline curves is sub-divided so that two spline curves will be found that match the path of the original one. This is done iteratively until all paths are defined in terms of equal-duration splines, all of which share a common knot vector and parameter range. For example, for curves, $\boldsymbol{r}_1(\tau)$ and $\boldsymbol{r}_2(\tau)$, with durations $T_1 T_2$, the path with duration, T_2 is sub-divided. As noted earlier, the subdivision process is deferred in cases that would produce segments shorter than 10 s in duration.[e] To determine the new spline control points, the curve with duration T_2 is sampled at equally spaced parametric intervals from 0 to $\tau_{\text{cut}} \,|\, S_2(\tau_{\text{cut}}) = S_1(\tau_{\max}) V_2/V_1$ and again at equally spaced parametric intervals from τ_{cut} to $\tau_{\max}$. This step yields two sets of points on the longer duration curve that are interpolated to determine the subdivided curves. Direct interpolation of points from the original curve is used for computational expediency. Twelve points are needed for each segment to maintain a constant number of control points. In order to ensure smooth subdivision, the parameter assignment for these points *with respect to the new splines* is taken to be the values, where each basis function reaches its maximum value. This provides a balanced weighting to all points and tends to minimize any discontinuities at the junctures that arise because the solution is based solely on position. Details are presented in Appendix D.

7. Flight Test Configuration

The NPS UAV lab has developed a Rapid Flight Test Prototyping System (RFTPS) to enable on-board integration of advanced control algorithms from concept to flight test. The principal capability of the system enables advanced control development, integration and deterministic execution of the resulting code on-board, and advanced data logging and communication capabilities. The system is based on tight integration of several software and hardware components including convenient Matlab/Simulink/RTW research and development environment, an advanced industrial autopilot and a high performance microcomputer. Major benefits of the developed RFTPS system include:

- convenience of high level algorithm design, auto-coding capabilities and hard real-time execution,
- simple and transparent interfacing of hardware that is primarily based on the well supported serial and TCP/IP communication,
- inexpensive design of on-board components based on COTS technology,
- advanced payloads, which are highly reliable, using industrial grade PC104 boards and miniature sensors that can be easily changed or upgraded as mission requirements dictate; the telemetry links are secure, yet

[d] Equal path length for common constant velocity paths.

[e] The constraint that no paths be shorter than 10 s in duration is imposed as a practical consideration to avoid rapid route switching that can induce large coordination errors.

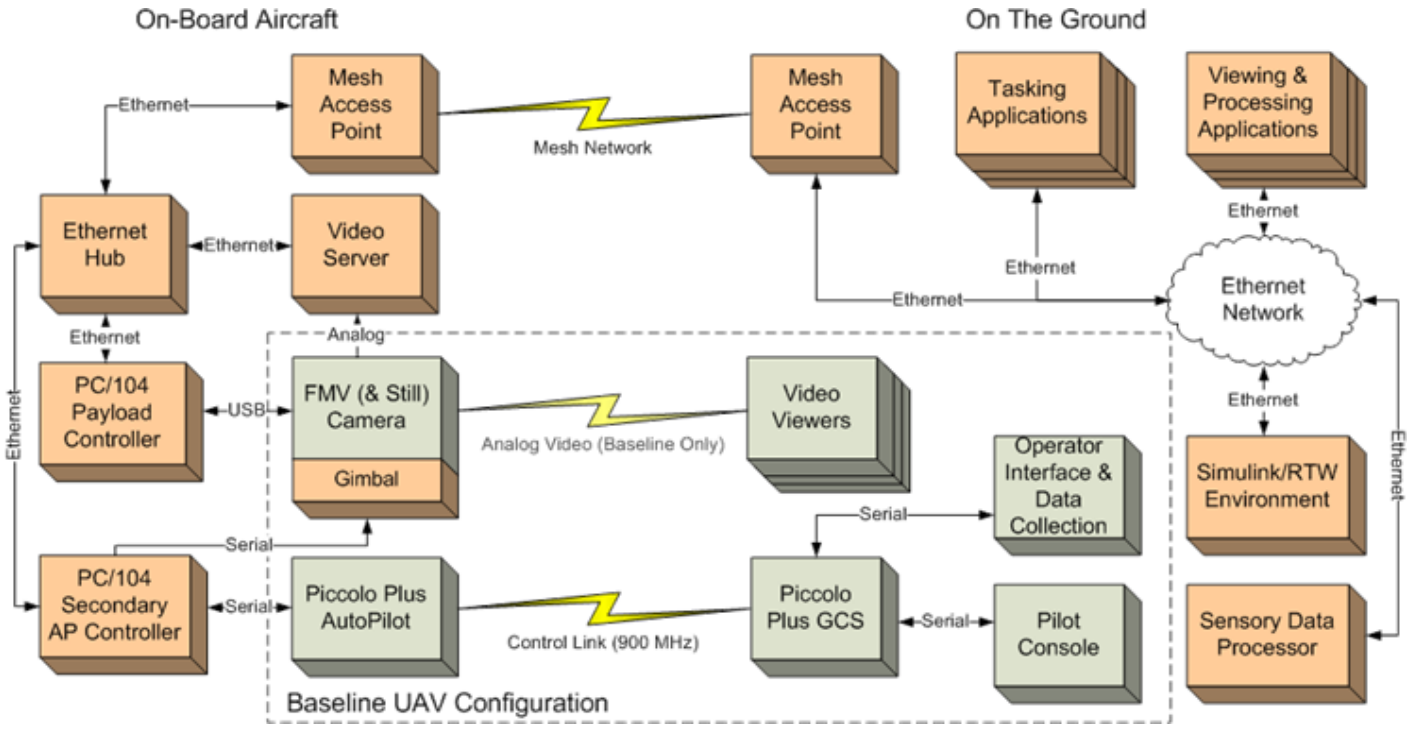

Fig. 7. (Color online) NPS UAV rapid flight control prototyping system overview.

low power and unobtrusive to the public, thus dispensing with the need for special authorizations from government authorities.

An overview of the system configuration is presented in Fig. 7. A key feature of this system is the integration of all payload subsystems via Ethernet. On-board the vehicle, subsystems are tied to a mesh access point through which data are communicated to the ground and other cooperative UAVs. The IP-based architecture makes the system scalable to any sensor configuration that can be carried by the vehicle and provides open and secure access. Moreover, the self-configuring wireless mesh network data communication (WaveRelay) is independent from the primary autopilot command/control link for safety of flight. The mission sensors used for this research include full motion video (FMV) and still cameras that are pointed by gimbal systems with one (roll stabilized) or two degrees of freedom (pan-tilt for direct pointing).

At the flight test site, a mobile ground control station permits full access to RFTPS features (Fig. 8).

The RFTPS has been demonstrated on both conventional fixed-wing propeller driven and powered glider configurations, as shown on Fig. 9. The Rascal UAV has been the test UAV for this research.

Prior to flight test, algorithms and hardware are thoroughly checked in a simulation environment. Our validation and verification process features three major steps (i) nonlinear, software-based, simulation (ii) Hardware-in-the-loop (HITL) simulation and (iii) flight test. First, the nominal aircraft is tested in comprehensive nonlinear simulations. The results are then transitioned into multi-vehicle HITL simulation, including radio links and sensors. Finally, the viable scenarios derived at the simulations phase are executed in flight test.

Fig. 8. (Color online) Mobile ground control station.

Fig. 9. NPS autonomous vehicles available to validate this research: Rascal (left panel), Glider (right panel).

8. Example Scenario

The preceding route finding and spline approximation algorithms were flight tested in August 2012 at a site used by the Naval Postgraduate School for UAV experiments. Results of these experiments confirmed the value of encoding trajectory definitions into parametric equations to minimize communication requirements, especially when dynamically re-routing vehicles. Flight tests of coordinated constant duration subdivided splines were then accomplished in February 2013 at the same site. Data from a typical scenario involving two UAVs and multiple sub-goals from those tests is illustrated in Fig. 10. The red polygon outlines the chosen UAV range at Camp Roberts, CA. The paths selected by the planner are described in Sec. 4 are shown in red and blue. The overall performance of our entire system executing these paths as defined in Sec. 6.3 is shown in Fig. 11. Track offsets from the plan due to strong winds from the Northeast is evident, underscoring the importance of reserving speed control authority for disturbance rejection. The precision of vehicle trajectory coordination can be seen in Fig. 12, which shows a time

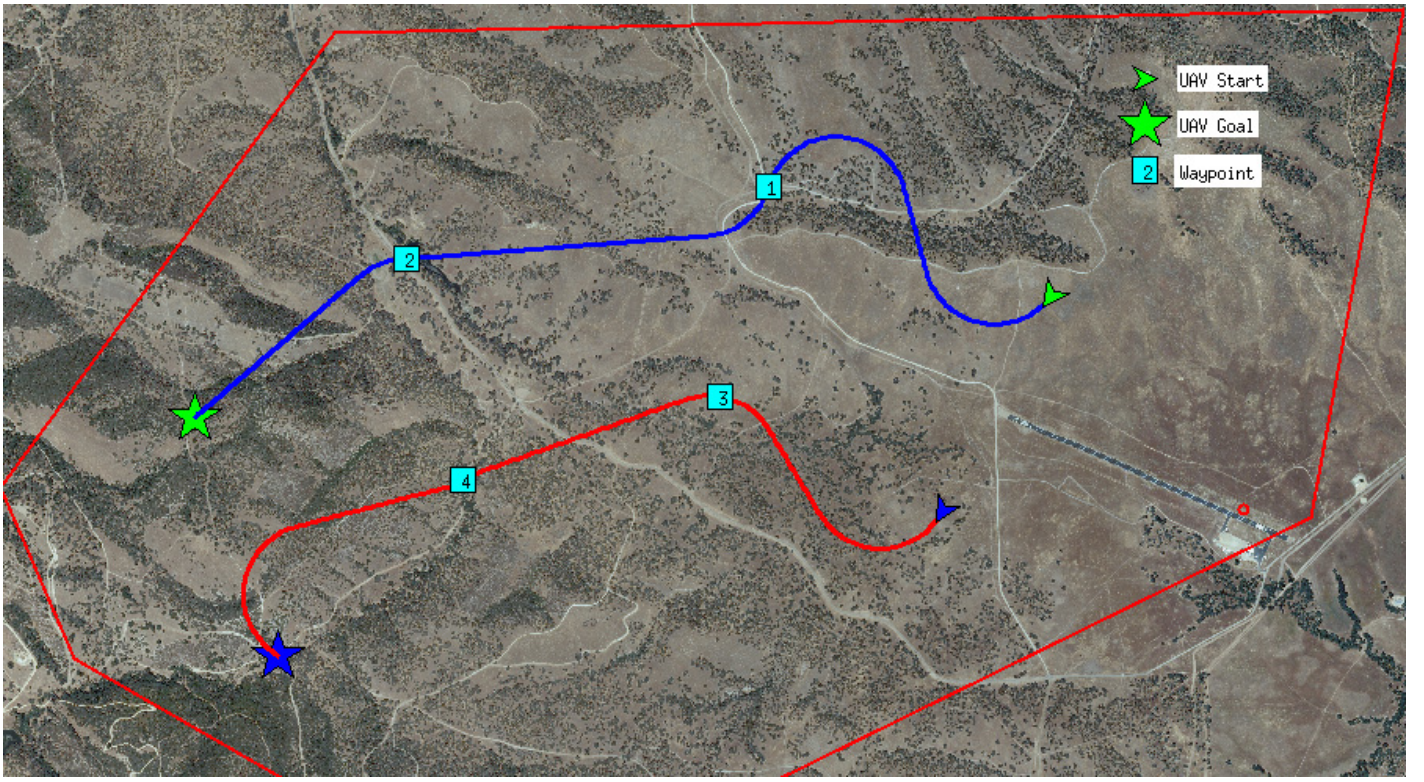

Fig. 10. (Color online) Coordinated mission profiles for two UAVs — as defined by the planner.

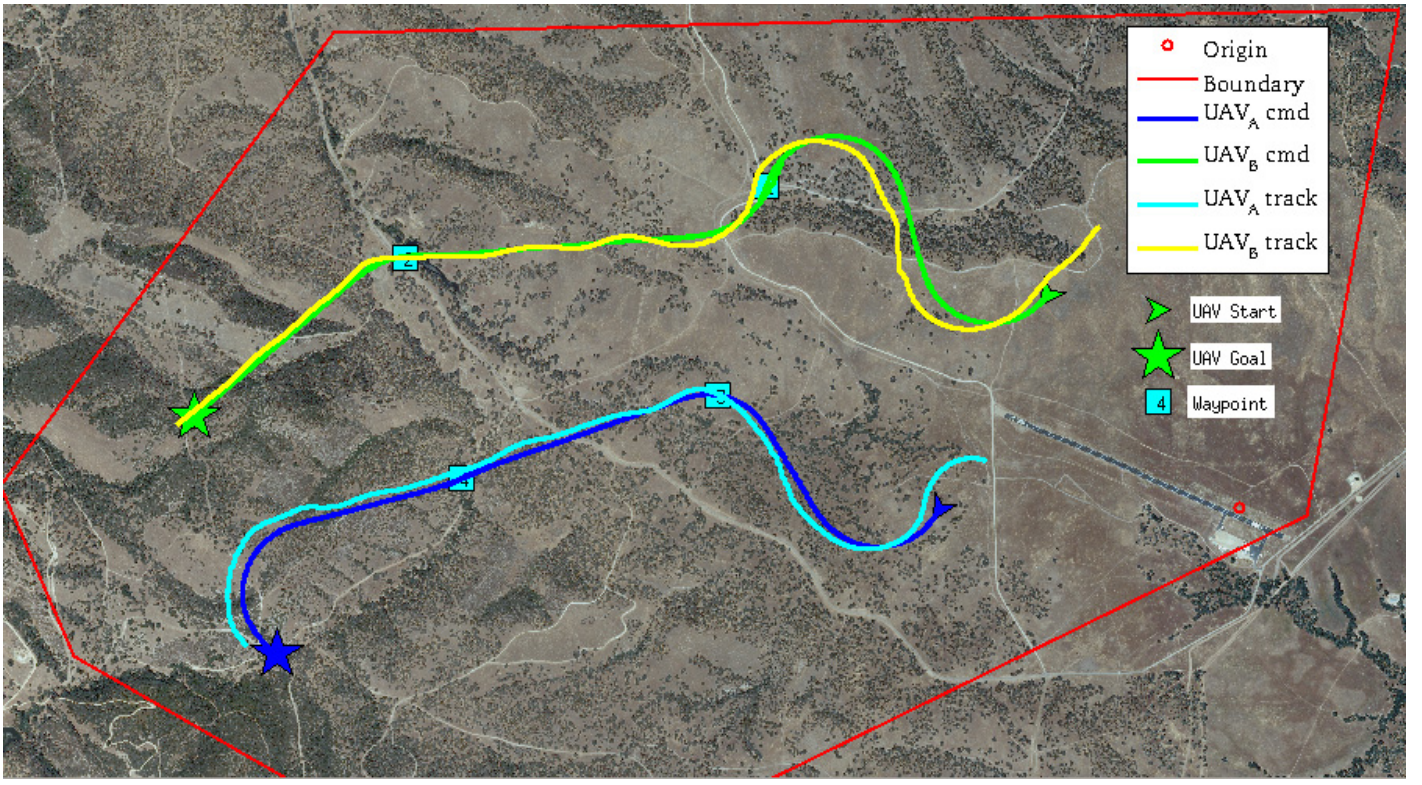

Fig. 11. (Color online) Coordinated mission profiles for two UAVs — as executed in strong winds.

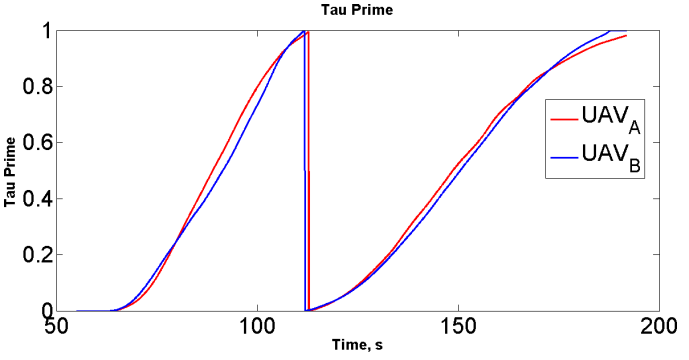

Fig. 12. (Color online) Vehicle coordination time history across two typical segments of a flight plan.

history of the parametric progress of each vehicle across two segments of the flight plan. In this figure, Tau-prime is the normalized parameter used to define the location of the vehicle with respect to starting and ending points of each segment. Two segments are shown to permit the figure to be drawn with high resolution. These results are typical of what was demonstrated.

9. Conclusion

Precision path following is key performance requirement for reconnaissance or surveillance type missions. We have developed a set of algorithms that transcribes a set of individual points into a single continuous path using a compact definition via B-splines. In this manner, communication bandwidth and on-board computational requirements for vehicle assignment and navigation can be minimized. We demonstrated that polynomial basis functions of degree 4 are the lowest order that can permit abstraction of the dynamic constraints in this application. The B-spline parametric curve formulation provides robust fidelity to the original plans and is structured to provide feasible paths that account for dynamic response constraints. Since real-time trajectory interrupt and redirection are also important capabilities for these types of missions to be responsive to real-world developments, we leveraged the flexibility provided by B-splines to include these features. The parametric curve formulation was then integrated with multi-vehicle coordination. Through flight testing in February 2013, we found that path segments shorter than 10 s in duration are not practical with respect to coordination. As noted in

Sec. 6.1, a remaining direction of research we are pursuing is to leverage the flat dynamics approach to derive the paths directly in the B-spline domain. In this domain, adherence to rate of change of curvature constraints poses the most significant computational complexity challenge.

Acknowledgments

We would like to acknowledge Jeff Wurz and the Joint Interagency Field Exploration organizers at the Naval Postgraduate School who enabled us to validate these algorithms in a flight test setting at the Camp Roberts facility in the California central coastal hills. Furthermore, we gratefully acknowledge support from the U.S. Office Office of Noval Research (ONR) grants N00014-09-1-1031 and 10936907.

Appendix A. Computation of the B-Spline Basis Functions

B-spline basis functions are recursively defined over a knot sequence. Some use a convention that the curve parameter and knot sequence are drawn from the domain $[0, 1]$. However, while zero is a convenient starting point, it is only important that both are drawn from the same domain and that the knot sequence terminate at the maximum parameter value. In general, the curve parameter, τ, and knot sequence, τ_i, $i = 0 \ldots d + 1 + M$. are defined:

$$\begin{aligned} \tau &\in [0, \tau_{\max}], \\ 0 &= \tau_0 \le \tau_i \le \tau_{i+1} \le \tau_m \\ &= \tau_{\max}. \end{aligned} \tag{A.1}$$

The basis functions are then generated recursively:

$$\begin{aligned} N_{i,d}(\tau) = &\frac{\tau - \tau_i}{\tau_{i+d} - \tau_i} N_{i,d-1}(\tau) \\ &+ \frac{\tau_{i+d+1} - \tau}{\tau_{i+d+1} - \tau_{i+1}} N_{i+1,d-1}(\tau), \end{aligned} \tag{A.2}$$

starting with:

$$N_{i,0}(\tau) = \begin{cases} 1 & \text{if } \tau \in [\tau_i, \tau_{i+1}] \\ 0 & \text{otherwise} \end{cases}.$$

Note, to avoid division by zero when computing Eq. (A.2), $N_{i,d}(\tau) = 0$ when the denominator of either term in Eq. (A.2) vanishes. Rather than writing recursive code to generate $N_{i,4}$ basis functions, we use a matrix form of Eq. (A.2) as derived by Ref. 14. The matrix form can be derived by noting that over a nonempty knot span a single basis function spans its degree +1. So, $N_{i,0}(\tau)$ is nonzero only on the span τ_i through τ_{i+1}. Similarly, $N_{i,1}(\tau)$ is nonzero only on the span τ_i through τ_{i+2} and so forth, so that a function of degree d is nonzero over a span of $d + 1$ knots. Given a knot vector that spans $\tau_0 = 0 \ldots \tau_{\max} = M + 1 - d$, the values of these functions at an arbitrary parameter value can be recursively computed in matrix form starting at $N_{i,1}(\tau)$. The values of the *two* $N_{i,1}(\tau)$ basis functions can be expressed as the row vector of the following matrix equation:

$$[N_{i-1,1}(\tau), N_{i,1}(\tau)] = \left[\frac{\tau_{i+1} - \tau}{\tau_{i+1} - \tau_i}, \frac{\tau - \tau_i}{\tau_{i+1} - \tau_i}\right]. \tag{A.3}$$

The values of the *three* $N_{i,2}(\tau)$ basis functions can be expressed as the row vector result of this matrix equation:

$$\begin{aligned} &[N_{i-2,2}(\tau), N_{i-1,2}(\tau), N_{i,2}(\tau)] \\ &\quad = [N_{i-1,1}(\tau), N_{i,1}(\tau)] \\ &\qquad \times \begin{bmatrix} \dfrac{\tau_{i+1} - \tau}{\tau_{i+1} - \tau_{i-1}} & \dfrac{\tau - \tau_{i-1}}{\tau_{i+1} - \tau_{i-1}} & 0 \\ 0 & \dfrac{\tau_{i+2} - \tau}{\tau_{i+2} - \tau_i} & \dfrac{\tau - \tau_i}{\tau_{i+2} - \tau_i} \end{bmatrix}. \end{aligned} \tag{A.4}$$

Adding another degree to the basis polynomials adds another function to those spanning any consecutive knot span, so the row vector of function values at parameter, τ expands by one. This structure permits spline functions of any degree, d, to be computed as a matrix product as follows:

$$N_{i,d}(\tau) = \boldsymbol{R}_1(\tau)\boldsymbol{R}_2(\tau) \ldots \boldsymbol{R}_d(\tau). \tag{A.5}$$

Thus, the dimension of the matrix $\boldsymbol{R}_k$, for each integer $k \le d$ is $[k \times k + 1]$. The general form for $\boldsymbol{R}_k(\tau)$ is block diagonal with only two nonzero entries on each row starting at the diagonal element. The matrix equation for these two elements between knots τ_i and τ_{i+1} at indices (n, n) and $(n, n + 1)$ is:

$$\begin{aligned} &\boldsymbol{R}_k(\tau)_{(n,n:n,n+1)} \\ &\quad = \begin{bmatrix} \dfrac{\tau_{i+n} - \tau}{\tau_{i+n} - \tau_{i+n-k}} & \dfrac{\tau - \tau_{i+n-k}}{\tau_{i+n} - \tau_{i+n-k}} \end{bmatrix}. \end{aligned} \tag{A.6}$$

Since these functions are intended for parametric trajectories, there must be a basis function for each control point in the spline. For minimum computational complexity with favorable numerical stability, we use even integer knot spacing for our splines with clamped endpoints. The minimum length knot vector is, therefore, $2(d + 1)$, for which only starting and ending points are defined. For $M + 1$ control points and degree d basis functions, our knot vectors have $M + d + 2$ elements, structured so the first $d + 1$ knots are at 0 and the last $d + 1$ knots are at $M + 1 - d$. Knots beginning at the index $d + 2$ are integers from 1 to $M + 1 - d$. For example, for six control points and quartic splines, the knot vector would be: $[0, 0, 0, 0, 0, 1, 2, 2, 2, 2, 2]$.

Appendix B. Computation of the B-Spline Arc-Length

As noted in Sec. 6.1, the irregular spacing of parameter versus arc-length along the curve mandates determination of this function for arbitrary values of arc-length. Firstly, we compute arc-length at each knot along the curve using a Gaussian quadrature formula as derived by Ref. 16.

$$\int_a^b f(\tau)d\tau = \sum_{j=1}^{n} H_j f(a_j) + E \tag{B.1}$$

where a_j are the specified zeros of the Legendre−Gauss quadrature polynomial of degree n. For n points, the highest degree polynomial for which the quadrature error, E can be made null is $2n-1$, so we select a three point formula for minimum complexity in our application. For an arbitrary value of τ between knots, the quadrature can be set up over the range of 0 to 1, so the appropriate zero locations are: $[\frac{0.225403\hat{\tau}}{2}, \frac{\hat{\tau}}{2}, \frac{1.774597\hat{\tau}}{2}]$, where $\hat{\tau}$ is the minimum remainder: $\hat{\tau} \equiv \min(\tau - \tau_i)|\hat{\tau} \geq 0$ (i.e., remainder with respect to nearest lower knot). The quadrature is executed across the entire knot vector to compute arc-length at each knot. Since we ultimately require the the precise parameter value associated with an arbitrary arc-length distance, S, we couple application of this quadrature formula with a Newton's Method algorithm, as suggested by Ref. 15 to find the root of

$$L(\tau) - S = 0. \tag{B.2}$$

Since we are operating on a smooth polynomial to find the appropriate parameter, τ, only three iterations are required if we start from the nearest last known pair $(\tau, L(\tau))$. The quadrature function is called each iteration. To ensure the Newton's Method does not erroneously overshoot into the adjacent knot span, we bound the minimum and maximum values an iteration can attain. In this case, the Newton's Method iterations can be expressed:

$$\tau_{j+1} = \tau_j - \frac{L(\tau_j) - S}{\|L'(\tau_j)\|}, \tag{B.3}$$

$\|L'(\tau_j)\| \equiv$ magnitude of parametric speed at τ_j.

In order to compute the parametric speed, we use the matrix equation for the basis functions and, using D as the operator for differentiation of a matrix in this case with respect to the parameter τ, note:

$$DN_{i,d}(\tau) = dN_{i,d-1}(\tau)D\boldsymbol{R}_d. \tag{B.4}$$

The form of the matrix of derivatives, $D\boldsymbol{R}_d$, is also block diagonal with the same dimension as $\boldsymbol{R}_d$. It also has two nonzero elements on each row. The matrix equation for these two elements between knots τ_i and τ_{i+1} at indices (n, n) and $(n, n+1)$ is:

$$D\boldsymbol{R}_d(\tau)_{(n,n:n,n+1)} = \left[\frac{-1}{\tau_{i+n} - \tau_{i+n-d}} \quad \frac{1}{\tau_{i+n} - \tau_{i+n-d}}\right]. \tag{B.5}$$

Using these equations, a function can be written that uses Newton's Method to identify the parameter, τ, associated with a specific arc-length along the curve. The distance, knot vector, vector of arc-length distances associated with each knot, and the B-spline curve control points are used as input to return the associated parameter value.

Appendix C. Least Squares Approximation Algorithm for B-Splines

Given a set of points, $\boldsymbol{q}_l$ $l = 0 \ldots L$, the B-spline curve that best approximates the set can be determined:

The parametric spline curve is defined:

$$\boldsymbol{r}(\tau) = \sum_{m=0}^{M} N_{m,d}(\tau)\boldsymbol{p}_m,$$

where the elements of a column vector, $\boldsymbol{P}$,

of control points are defined : $\boldsymbol{p}_m \,|\, m = 0 \ldots M.$

The control points derived from boundary conditions are :

$\boldsymbol{p}_0 = q_0,\ \boldsymbol{p}_M = q_L.$

The internal control points are derived from

$$\text{a minimization : } \min \sum_{l=1}^{L-1} \left(\boldsymbol{q}_l - \sum_{m=1}^{M-1} N_{m,d}(\tau_l)\boldsymbol{p}_m\right)^2. \tag{C.1}$$

A matrix of basis function values (for the functions that multiply the internal control points, $N_{1,d}(\tau_l)$ through $N_{M-1,d}(\tau_l)$) is computed for the internal points ($\boldsymbol{q}_1$ through $\boldsymbol{q}_{L-1}$ in the set to be approximated.

$$\boldsymbol{N} = \begin{bmatrix} N_{1,d}(\tau_1) & N_{2,d}(\tau_1) & \ldots & N_{M-1,d}(\tau_1) \\ N_{1,d}(\tau_2) & N_{2,d}(\tau_2) & \ldots & N_{M-1,d}(\tau_2) \\ \vdots & \vdots & \ddots & \vdots \\ N_{1,d}(\tau_{L-1}) & N_{2,d}(\tau_{L-1}) & \ldots & N_{M-1,d}(\tau_{L-1}) \end{bmatrix}. \tag{C.2}$$

The constant terms in Eq. (C.1) are also grouped into a column vector, $\boldsymbol{Q}$, for each internal point in the set to be

approximated.

$$\boldsymbol{Q} = \begin{bmatrix} q_1 - N_{0,d}(\tau_1)q_0 - N_{M,d}(\tau_1)q_L \\ q_2 - N_{0,d}(\tau_2)q_0 - N_{M,d}(\tau_2)q_L \\ \vdots \\ q_{L-1} - N_{0,d}(\tau_{L-1})q_0 - N_{M,d}(\tau_{L-1})q_L \end{bmatrix}. \tag{C.3}$$

Now, the function to be minimized in Eq. (C.1) can be expressed:

$$\min \sum_{l=1}^{L-1} \left(\sum_{m=1}^{M-1} N_{m,d}(\tau_l)\boldsymbol{p}_m - \boldsymbol{Q}_l \right)^2. \tag{C.4}$$

The solution for the column vector of internal control points, $\boldsymbol{P}$, can be expressed[17]:

$$\begin{aligned} \boldsymbol{R} &= \boldsymbol{N}^T\boldsymbol{Q}, \\ (\boldsymbol{N}^T\boldsymbol{N})\boldsymbol{P} &= \boldsymbol{R}. \end{aligned} \tag{C.5}$$

In order to ensure the matrix $\boldsymbol{N}^T\boldsymbol{N}$ is not singular, there must be at least one data point in the set to be approximated within each knot span, τ_i to $\tau_{i+1} \,|\, i = 0 \ldots M + d + 2$. Our planner satisfies this assumption.

Appendix D. Derivation of B-Spline Control Points for Matched Duration Splines

In order to determine the control points associated with each portion of a subdivided spline curve, we first determine two sets of 12 points along the curve. The first set, $\boldsymbol{Q_1}$, spans from the starting point to the arc length distance that defines where the division is made and the second set, $\boldsymbol{Q_2}$, starts at the division point and spans the remainder of the original path. The first set is computed using control points, $\boldsymbol{P}$, and parameters: $\tau_j \,|\, j = 1:12$, where $\tau_1 = 0$ and $\tau_{12} = \tau_{\text{cut}}$:

$$\begin{bmatrix} \boldsymbol{Q_{1_1}} \\ \boldsymbol{Q_{1_2}} \\ \vdots \\ \boldsymbol{Q_{1_{12}}} \end{bmatrix} = \begin{bmatrix} N_{1,4}(\tau_1) & N_{2,4}(\tau_1) & \ldots & N_{12,4}(\tau_1) \\ N_{1,4}(\tau_2) & N_{2,4}(\tau_2) & \ldots & N_{12,4}(\tau_2) \\ \vdots & \vdots & \ddots & \vdots \\ N_{1,4}(\tau_{12}) & N_{2,4}(\tau_{12}) & \ldots & N_{12,4}(\tau_{12}) \end{bmatrix} \begin{bmatrix} \boldsymbol{P_1} \\ \boldsymbol{P_2} \\ \vdots \\ \boldsymbol{P_{12}} \end{bmatrix}. \tag{D.1}$$

The second set of points for interpolation is found in a similar fashion using the same control points but spline basis functions evaluated over the parametric range starting at τ_{12} through $\tau_{\max}$. To determine the two new sets of

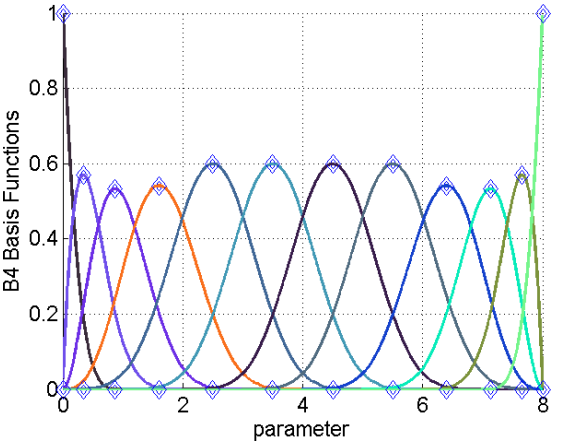

Fig. A.1 (Color online) Quartic B splines defined for *clamped* uniform knots over the range 0 to 8 (12 control points interpolated at parameters indicated with diamond symbols).

control points, $\boldsymbol{P'_1}$ and $\boldsymbol{P'_2}$, we use the same clamped uniform knot sequence to form basis functions but we evaluate the functions for interpolation at precomputed values defined by the maximum of each basis function. These points are shown with a diamond symbol on Fig. A.1. In this manner, the weighting of each basis function is relatively uniform. The interpolation of two new B-spline definitions can then be expressed:

$$\begin{bmatrix} \boldsymbol{Q_{1_1}} \\ \boldsymbol{Q_{1_2}} \\ \vdots \\ \boldsymbol{Q_{1_{12}}} \end{bmatrix} = \begin{bmatrix} N_{1,4}(\tau_1) & N_{2,4}(\tau_1) & \ldots & N_{12,4}(\tau_1) \\ N_{1,4}(\tau_2) & N_{2,4}(\tau_2) & \ldots & N_{12,4}(\tau_2) \\ \vdots & \vdots & \ddots & \vdots \\ N_{1,4}(\tau_{12}) & N_{2,4}(\tau_{12}) & \ldots & N_{12,4}(\tau_{12}) \end{bmatrix} \begin{bmatrix} \boldsymbol{P'_1} \\ \boldsymbol{P'_2} \\ \vdots \\ \boldsymbol{P'_{12}} \end{bmatrix}, \tag{D.2}$$

where $\tau_{1\ldots12}$ are defined as $[0, 0.3408, 0.872, 1.605, 2.5, 3.5, 4.5, 5.5, 6.395, 7.128, 7.6585, 8]$, as shown on Fig. A.1. In compact notation, this can be expressed:

$$\boldsymbol{Q_1} = \boldsymbol{N'}\boldsymbol{P'_1}. \tag{D.3}$$

The final equation for the control points is then simply:

$$\boldsymbol{P'_1} = (\boldsymbol{N'})^{-1}\boldsymbol{Q_1}. \tag{D.4}$$

The second set of control points, $\boldsymbol{P'_2}$, are found in a similar fashion.

References

[1] I. J. Schoenberg, Contributions to the problem of approximation of equidistant data by analytic functions, *Quart. Appl. Math.* **8** (1946) 45–99, 112–141.

[2] C. de Boor, *A Practical Guide to Splines* (Springer-Verlag, New York, 1978).
[3] E. Seckel, *Stability and Control of Airplanes and Helicopters* (Academic Press, New York, 1964).
[4] R. Stengel, *Flight Dynamics* (Princeton University Press, Princeton NJ, 2004).
[5] P. E. Hart, N. J. Nilsson and B. Raphael, A formal basis for the heuristic determination of minimum cost paths, *IEEE Trans. Syst. Sci. Cybernet. SSC* **4**(2) (1968) 100−107.
[6] E. Xargay, V. Dobrokhodov, I. Kaminer, A. M. Pascoal, N. Hovakimyan and Chengyu Cao, Time-critical cooperative control of multiple autonomous vehicles: Robust distributed strategies for path-following control and time-coordination over dynamic communications networks, *Cont, Syst. IEEE* **32**(5) (2012) 49−73.
[7] R. L. Bishop, There is more than one way to frame a curve, *The Amer. Math. Monthly* **82**(3) (1975) 246−251.
[8] R. Murray, Recent research in cooperative control of multivehicle systems, *J. Dynam. Syst. Measure. Control* **129** (2007) 571−583.
[9] E. Kharisov, X. Wang and N. Hovakimyan, Distributed event-triggered consensus algorithm for uncertain multi-agent systems Toronto, Canada, August 2010. AIAA-2010-8325.
[10] E. Xargay, I. Kaminer, A. Pascoal, N. Hovakimyan, V. Dobrokhodov, V. Cichella, A. P. Aguiar and R. Ghabcheloo, Time-critical cooperative path following of multiple uavs over time-varying networks, (2013) (in press).
[11] Ph. Martin, R. M. Murray and P. Rouchon, Flat systems, equivalence and trajectory generation, Technical report, Caltech Control and Dynamical Systems Technical Reports, Caltech, Pasadena, CA (2003).
[12] Ph. Martin, R. M. Murray and P. Rouchon, Flat systems, equivalence and trajectory generation, Technical Report, California Institute of Technology (2003).
[13] R. Farouki and T. Sakkalis, Real rational curves are not unit speed, *Comput. Aided Geomet. Desi* **8** (1991) 151−157.
[14] T. Lyche and K. Mørken, *Spline Methods Draft*, Department of Informatics, Centre of Mathematics for Applications, University of Oslo (2011).
[15] J. W. Peterson, Arc length parameterization of spline curves. Available at http://crtl-i.com/blog/wp-content/uploads/2007/11/re-param.pdf (2007).
[16] A. Ralston, *A First Course in Numerical Analysis*, 2nd edn. (McGraw-Hill, 1978).
[17] Y. M. Li, Least squares approximation, Intergraph. Available at http://www.xxx.pdf (1997).