Real-Time Dynamic Trajectory Smoothing for Unmanned Air Vehicles

Erik P. Anderson, Randal W. Beard, and Timothy W. McLain

Abstract—**This brief presents a real-time, feasible trajectory generation algorithm for unmanned air vehicles (UAVs) flying through a sequence of waypoints. The algorithm produces extremal trajectories that transition between straight-line path segments in a time-optimal fashion. In addition, the algorithm can be configured so that the dynamically feasible trajectory has the same path length as the straight-line waypoint path. Implementation issues associated with the algorithm are described in detail. Simulation studies show the effectiveness of the proposed method.**

Index Terms—**Autonomous systems, optimal control, path planning, trajectory generation, unmanned air vehicles (UAVs).**

I. Introduction

RECENT advances in computing, sensing, and battery technology have made unmanned vehicles a viable option in both military and commercial applications [1]. For unmanned vehicle technology to be useful, autonomous, semiautonomous, and cooperative behaviors must be developed. Central to these behaviors are automatic trajectory generation algorithms.

Our particular interest has been cooperative timing problems for unmanned air vehicles (UAVs) [2], [3], where the trajectories must be both feasible, and satisfy stringent timing or path length constraints. Our approach decomposes the trajectory generation problem into two steps: a waypoint path planning step, where the straight-line paths are not time-parameterized, and a time-parameterized trajectory generation step to "smooth" the waypoint paths into dynamically feasible trajectories [4]. This brief describes the trajectory generation procedure. The salient features of our approach are that the trajectory generator: 1) smoothes through a set of waypoints, minimizing the deviation from the associated waypoint path; 2) satisfies the curvature and velocity constraints imposed by the dynamics of the UAV; 3) maintains the path length of the associated waypoint path; and 4) operates in real-time.

While the trajectory generation technique introduced in this brief is well suited to cooperative timing missions, it can also be used in problems where timing may not be critical. For example, it may be desirable to pass directly over the waypoints while minimizing the deviation from the original waypoint path. Another possible scenario is that the UAV could carry a sensor with a finite-dimensional footprint [5]. It may be desired to configure the trajectory generator so that the UAV passes within a specific distance of the waypoint while minimizing the transition time between the straight-line path segments. The technique presented in this brief is well matched to these scenarios.

Our approach is based on the local reachability region of the aircraft and on basic geometry. In that sense, it is similar to the approaches reported in [2], [6], and [7]. In [6], it is shown that the shortest path between two points satisfying curvature constraints comprises circles and straight-line path segments. Reference [7] builds upon Dubins's ideas to generate feasible trajectories for UAVs given kinematic and path constraints by algorithmically finding the optimal location of Dubins circles and straight-line paths. In [2], Dubins circles are superimposed as fillets at the junction of straight-line waypoint paths produced from a Voronoi diagram. Our approach differs from [2], [6], and [7] in that the trajectory is generated in real-time by a dynamic process. Rather than inserting fillets and planning the trajectory *a priori*, our approach dynamically generates the trajectory in flight. The advantage of doing this is that the algorithm is more reactive to dynamically changing environments and temporally distributes the computations, making it feasible to implement in real-time.

Several other approaches to UAV path planning have recently appeared in the literature including probabilistic maps [8], differential flatness [9], [10], and optimal control techniques [11], [12].

II. Problem Statement and Preliminaries

We will assume that the UAV is flying at a constant altitude and that the UAV dynamics subject to velocity-hold and heading-hold autopilots are first order [13]

$$\dot{z}_x = v\cos\psi \tag{1}$$
$$\dot{z}_y = v\sin\psi \tag{2}$$
$$\dot{\psi} = \alpha_\psi(\psi^c - \psi) \tag{3}$$
$$\dot{v} = \alpha_v(v^c - v) \tag{4}$$

where α_ψ and α_v are known constants that depend on the implementation of the autopilot. In addition, the underlying UAV dynamics constrain the heading rate and velocity as follows:

$$-c \le \dot{\psi} \le c \tag{5}$$
$$0 < v_{\min} \le v \le v_{\max}. \tag{6}$$

Definition 1: A trajectory $\mathbf{z}^d(t) = (z_x^d, z_y^d)^\top$ is called *dynamically feasible* if there exist inputs $\psi^c(t)$ and $v^c(t)$ such that $\mathbf{z}(0) = \mathbf{z}^d(0)$ implies that $\mathbf{z}(t) = \mathbf{z}^d(t)$, and the dynamics (1)–(4) and constraints (5)–(6) are satisfied, for all $t \ge 0$.

The input to the trajectory generator is a waypoint path $\mathcal{P} = \{v, \{\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_N\}\}$ where $v \in [v_{\min}, v_{\max}]$ is the desired velocity of the UAV, and $\mathbf{w}_i \in \mathbb{R}^2$ denote the waypoints expressed in inertial coordinates.

Manuscript received February 7, 2003; revised August 8, 2003. Manuscript received in final form July 16, 2004. Recommended by Associate Editor D. A. Schoenwald. This work was supported in part by the Air Force Office of Scientific Research under Grants F49620-01-1-0091 and F49620-02-C-0094, and in part by the Defense Advanced Research Projects Agency under Grant NBCH1020013.

E. P. Anderson is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305 USA (e-mail: eandersn@stanford.edu).

R. W. Beard is with the Electrical and Computer Engineering Department, Brigham Young University, Provo, UT 84602 USA (e-mail: beard@ee.byu.edu).

T. W. McLain is with the Mechanical Engineering Department, Brigham Young University, Provo, UT 84602 USA (e-mail: tmclain@et.byu.edu).

Digital Object Identifier 10.1109/TCST.2004.839555

In our approach, the trajectory generator is given by

$$\dot{\hat{z}}_x = \hat{v}\cos\hat{\psi} \tag{7}$$
$$\dot{\hat{z}}_y = \hat{v}\sin\hat{\psi} \tag{8}$$
$$\dot{\hat{\psi}} = u \tag{9}$$

where $u \in [-c, c]$ is an input that will be selected to meet the specified objectives. Note that (7)–(9) have a mathematical structure similar to the kinematics of the UAV. We will assume that the trajectory is to be traversed at a constant velocity $\hat{v} \in [v_{\min}, v_{\max}]$. Note that if $\hat{\mathbf{z}}(0) = \mathbf{z}(0)$, then (7)–(9) are guaranteed to generate dynamically feasible trajectories. Therefore the feasibility issue is satisfied *a priori*. Equations (7)–(9) are solved via a fixed-step ordinary differential equation (ODE) solver and propagated in real-time. In other words, the output of the trajectory smoother corresponds in time to the evolution of the UAV dynamics. If a fourth-order Runge–Kutta algorithm [14] is used, then u will need to be computed four times each sample period. Therefore, the computational complexity depends on the computation of u.

Note that if $u = +c$, then the trajectory smoother given in (7)–(9) traces out a right-handed circle whose center is given by

$$\mathbf{c}_R(t) = \hat{\mathbf{z}}(t) + R(-\sin(\hat{\psi}(t)) \quad \cos(\hat{\psi}(t)))^\top. \tag{10}$$

Similarly, if $u = -c$, then the trajectory smoother traces out the left-handed circle with center

$$\mathbf{c}_L(t) = \hat{\mathbf{z}} + R(\sin(\hat{\psi}(t)) \quad -\cos(\hat{\psi}(t)))^\top. \tag{11}$$

The local reachability region of the trajectory smoother is bounded by these two circles which have a radius given by $R = \hat{v}/c$. Note that as the desired velocity increases, the minimum turning radius increases. Since the boundaries of the reachability region are given by circles with known centers and radii, finding the intersection of the reachability region with lines and circles can be done in a computationally efficient manner.

III. Extremal Trajectories

In this section, we define a class of dynamically feasible trajectories called κ-trajectories, and show that they are minimum-time extremal trajectories.

Consider the waypoint path defined by the three waypoints $\mathbf{w}_{i-1}, \mathbf{w}_i$, and $\mathbf{w}_{i+1}$, and let

$$\mathbf{q}_i = (\mathbf{w}_{i-1} - \mathbf{w}_i)/\|\mathbf{w}_{i-1} - \mathbf{w}_i\|$$
$$\mathbf{q}_{i+1} = (\mathbf{w}_{i+1} - \mathbf{w}_i)/\|\mathbf{w}_{i+1} - \mathbf{w}_i\|$$

denote unit vectors along the corresponding path segments as shown in Fig. 1. Letting β denote the angle between $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$ we get $\beta = \cos^{-1}(\mathbf{q}_{i+1}^\top \mathbf{q}_i)$. As shown in Fig. 1, let $\bar{\mathcal{C}}$ be a circle of radius $R = \hat{v}/c$ whose center lies on the bisector of the angle formed by the three waypoints, such that the circle intersects both the lines $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$ and $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$ at exactly one point each. The bisector of β will intersect $\bar{\mathcal{C}}$ at two locations. Let $\bar{\mathbf{p}}$ be the intersection that is closest to $\mathbf{w}_i$.

From Fig. 1, it can be seen that $\sin(\beta/2) = R/(\|\bar{\mathbf{p}} - \mathbf{w}_i\| + R)$. Manipulating this expression, the distance between $\mathbf{w}_i$ and $\bar{\mathbf{p}}$ is found to be $\|\bar{\mathbf{p}} - \mathbf{w}_i\| = R(1/\sin(\beta/2) - 1)$. A unit vector

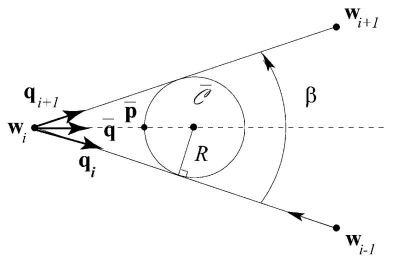

Fig. 1. Inscribed circle used in the definition in κ-trajectories.

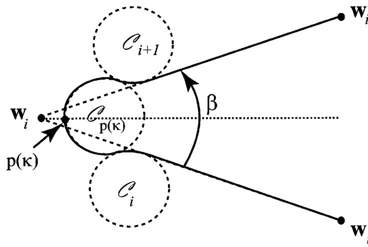

Fig. 2. Dynamically feasible κ-trajectory.

pointing along the bisector of the angle formed by the three waypoints is given by

$$\bar{\mathbf{q}} = \frac{\mathbf{q}_{i+1} - \mathbf{q}_i}{\|\mathbf{q}_{i+1} - \mathbf{q}_i\|}. \tag{12}$$

The point $\bar{\mathbf{p}}$ is therefore given by $\bar{\mathbf{p}} = \mathbf{w}_i + R(1/\sin(\beta/2) - 1)\bar{\mathbf{q}}$. Let $\mathbf{p}(\kappa)$ denote a parameterized point on the line between $\mathbf{w}_i$ and $\bar{\mathbf{p}}$, on the bisector of the angle

$$\mathbf{p}(\kappa) = \mathbf{w}_i + \kappa R\left(\frac{1}{\sin\left(\frac{\beta}{2}\right)} - 1\right)\bar{\mathbf{q}} \tag{13}$$

where $\kappa \in [0, 1]$. Clearly $\mathbf{p}(0) = \mathbf{w}_i$, and $\mathbf{p}(1) = \bar{\mathbf{p}}$.

As shown in Fig. 2, let $\mathcal{C}_{\mathbf{p}(\kappa)}$ be a circle of radius $R = \hat{v}/c$ whose center lies in the direction of $\bar{\mathbf{q}}$ and intersects $\mathbf{p}(\kappa)$. Also, let $\mathcal{C}_i$ be a circle of radius R placed such that it intersects $\mathcal{C}_{\mathbf{p}(\kappa)}$ and $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$ in exactly one location each. Define $\mathcal{C}_{i+1}$ similarly, as shown in Fig. 2.

Definition 2: A κ-trajectory is defined as the trajectory that is constructed by following the line segment $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$ until intersecting $\mathcal{C}_i$, which is followed until $\mathcal{C}_{\mathbf{p}(\kappa)}$ is intersected, which is followed until intersecting $\mathcal{C}_{i+1}$ which is followed until the line segment $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$ is intersected, as shown in Fig. 2.

Theorem 3: The κ-trajectory shown in Fig. 2 is the unique, dynamically feasible, minimum-time-extremal trajectory that transitions from the waypoint segment $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$ to the waypoint segment $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$, passing directly through $\mathbf{p}(\kappa)$.

Proof: The proof requires a change of variables followed by a standard application of Pontryagin's minimum principle [15]–[17]. ■

IV. Dynamic Trajectory Smoothing

This section describes a real-time algorithm that generates κ-trajectories for $\kappa \in [0, 1]$. If $\kappa = 1$, then the κ-trajectory transitions from the waypoint segment $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$ to the waypoint segment $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$ in minimum time. If $\kappa = 0$, then the κ-trajectory executes a minimum-time transition subject to the constraint that it passes directly over $\mathbf{w}_i$. We will also show in this

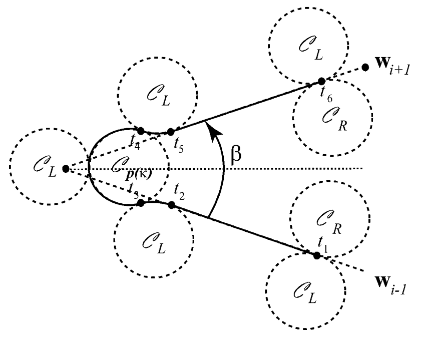

Fig. 3. Graphical depiction of the essential idea behind the selection of u.

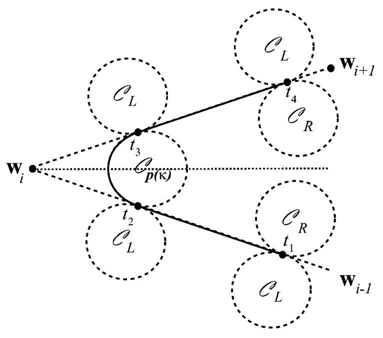

Fig. 5. Trajectory smoothing algorithm for $\kappa = 1$.

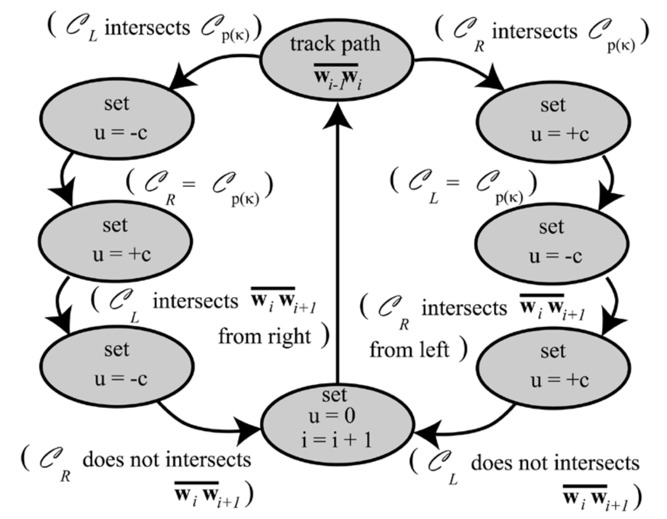
track path $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$
($\mathcal{C}_L$ intersects $\mathcal{C}_{p(\kappa)}$)
($\mathcal{C}_R$ intersects $\mathcal{C}_{p(\kappa)}$)
set u = -c
set u = +c
($\mathcal{C}_R$ = $\mathcal{C}_{p(\kappa)}$)
($\mathcal{C}_L$ = $\mathcal{C}_{p(\kappa)}$)
set u = +c
set u = -c
($\mathcal{C}_L$ intersects $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$ from right)
($\mathcal{C}_R$ intersects $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$ from left)
set u = -c
set u = +c
set u = 0 i = i + 1
($\mathcal{C}_R$ does not intersects $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$)
($\mathcal{C}_L$ does not intersects $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$)

Fig. 4. Block diagram of the trajectory smoothing algorithm for $\kappa \in [0, 1)$.

section how to select κ so that the κ-trajectory has the same path length as the original waypoint path.

The essential idea for the algorithm is shown in Fig. 3. The right and left turning constraints given in (10) and (11) are denoted by $\mathcal{C}_R$ and $\mathcal{C}_L$, respectively, at different time instances. The progression of time is denoted by $t_1, \ldots t_6$. The right turning constraint $\mathcal{C}_R$ is not shown at times t_2 and t_5 to avoid cluttering the figure. At time t_1 the trajectory smoother is tracking the waypoint segment $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$. When the left turning circle $\mathcal{C}_L$ intersects $\mathcal{C}_{\mathbf{p}(\kappa)}$ at time t_2, u is set to $-c$. The left turning constraint is followed until the right turning circle $\mathcal{C}_R$ corresponds exactly with $\mathcal{C}_{\mathbf{p}(\kappa)}$ at time t_3. The trajectory smoothing input u is then set to $+c$ and the right turning constraint is followed until the left turning constraint $\mathcal{C}_L$ intersects the waypoint segment $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$ at time t_4. The input u is again set to $-c$ until it reaches the waypoint segment at time t_5, where u is set to zero.

For $\kappa \in [0, 1)$, a flow chart of the trajectory smoothing selection scheme for u is shown in Fig. 4. The nominal state of the trajectory smoothing algorithm is to track the current waypoint path segment. Since the ODEs (7)–(9) are solved using a fixed-sample-rate solver, it is not possible to track the path by simply setting $u = 0$. A suitable tracking algorithm is discussed in Section V. When the trajectory smoother begins tracking the current waypoint segment, the location of $\mathcal{C}_{\mathbf{p}(\kappa)}$ is computed. If the turn is clockwise, then the constraint circle $\mathcal{C}_L$ is monitored until it intersects $\mathcal{C}_{\mathbf{p}(\kappa)}$, at which point $u \leftarrow -c$ (t_2 in Fig. 3). When $u = -c$, the motion of the trajectory smoother is such that $\mathcal{C}_L$ is stationary. The constraint circle $\mathcal{C}_R$ is monitored until it coincides exactly with $\mathcal{C}_{\mathbf{p}(\kappa)}$, at which time $u \leftarrow +c$ (time t_3). $\mathcal{C}_R$ is now stationary and $\mathcal{C}_L$ is monitored until it intersects the line segment $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$ from the right, at which time $u \leftarrow = -c$ (time t_4). The constraint circle $\mathcal{C}_L$ is stationary and $\mathcal{C}_R$ is monitored until it only intersects $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$ at one point, at which time tracking is resumed (time t_5). Similar steps are followed if the turn is counterclockwise.

The switching times are determined by finding circle and line intersections. Practical issues associated with finding the switching times in digital hardware are discussed in Section V.

If $\kappa = 1$, then the trajectory smoothing algorithm simplifies considerably. Similar to the case when $\kappa \in [0, 1)$, the first step is to determine $\mathcal{C}_{\mathbf{p}(\kappa)}$ and the direction of the turn. As shown in Fig. 5, for a clockwise turn the trajectory smoother tracks the straight-line path segment $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$ until $\mathcal{C}_R$ coincides with $\mathcal{C}_{p(\kappa)}$, at which point $u \leftarrow +c$ (t_2 in Fig. 5). $\mathcal{C}_R$ is then stationary and $\mathcal{C}_L$ is monitored until the entire circle is to the left of $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$ (time t_3), at which time tracking is resumed.

The following theorem asserts that the trajectory smoothing algorithm implements the extremal κ-trajectories defined in Section III.

Theorem 4: The trajectory smoothing algorithm depicted in Fig. 4, implements the extremal κ-trajectories defined in Section III.

Proof: See [16].

The trajectory smoother can be configured to run in several modes, depending on the application for which it is being used. For example, it may be desirable to choose κ so that the trajectory passes a distance D from the waypoint. This mode could be used to ensure that the footprint of a sensor attached to a UAV passes over the waypoint. If $D \in [0, R(1/\sin(\beta/2) - 1)]$, then using (13), it is straightforward to show that the proper choice of κ is

$$\kappa^* = \frac{D}{R}\frac{\sin\frac{\beta}{2}}{1 - \sin\frac{\beta}{2}}.$$

If it is desired that the trajectory pass directly over the waypoint, then choose $\kappa^* = 0$. On the other hand, if it is wished to transition between waypoints with only one turn, choose $\kappa^* = 1$.

For timing critical missions it is often desirable to plan the mission based on the waypoint paths. However, if the trajectory smoothing process changes the path length of the waypoint path, then the timing of the mission will be compromised. Therefore, it is desirable to choose κ such that the path length of the κ-trajectory is equal to the path length of the waypoint path. Toward that end, the next Lemma derives an analytic expression for the path length of a κ-trajectory.

Lemma 5: If $\kappa \in [0,1]$, and $R = \hat{v}/c$, then the path length of the κ-trajectory shown in Fig. 2 is given by

$$\begin{aligned}\mathcal{L}(\kappa, \mathbf{w}_{i+1}, \mathbf{w}_i, \mathbf{w}_{i-1}) &= \|\mathbf{w}_{i+1} - \mathbf{w}_i\| + \|\mathbf{w}_i - \mathbf{w}_{i-1}\| \\ &\quad + 2R\left(\frac{\pi - \beta}{2} + 2\cos^{-1}(\Xi(\kappa,\beta)) \right. \\ &\quad \left. - \Xi(\kappa,\beta)\sqrt{\frac{1}{\Xi(\kappa,\beta)^2} - 1} - (1-\kappa)\cos\frac{\beta}{2} - \kappa\cot\frac{\beta}{2}\right) \end{aligned} \tag{14}$$

where

$$\beta = \cos^{-1}\left(\left(\frac{\mathbf{w}_{i+1} - \mathbf{w}_i}{\|\mathbf{w}_{i+1} - \mathbf{w}_i\|}\right)^\top \left(\frac{\mathbf{w}_i - \mathbf{w}_{i-1}}{\|\mathbf{w}_i - \mathbf{w}_{i-1}\|}\right)\right) \tag{15}$$

and

$$\Xi(\kappa,\beta) = \frac{(1+\kappa) + (1-\kappa)\sin\frac{\beta}{2}}{2}. \tag{16}$$

Proof: The proof is an application in geometry [16], [17]. ■

From Lemma 5, it is clear that the path length difference between the waypoint path $\overline{\mathbf{w}_{i-1}\mathbf{w}_i\mathbf{w}_{i+1}}$ and the associated κ-trajectory is given by

$$\begin{aligned}\Lambda(\kappa, \mathbf{w}_{i+1}, \mathbf{w}_i, \mathbf{w}_{i-1}) &= 2R\left(\frac{\pi - \beta}{2} + 2\cos^{-1}(\Xi(\kappa,\beta)) \right. \\ &\left. - \Xi(\kappa,\beta)\sqrt{\frac{1}{\Xi(\kappa,\beta)^2} - 1} - (1-\kappa)\cos\frac{\beta}{2} - \kappa\cot\frac{\beta}{2}\right). \end{aligned} \tag{17}$$

The following lemma will be used to find κ such that the κ-trajectory has the same path length as the waypoint trajectory.

Lemma 6: If $\beta \in [0,\pi)$ and $\kappa \in [0,1]$, then Λ is a decreasing function of κ. In addition, $\Lambda(0, \mathbf{w}_{i+1}, \mathbf{w}_i, \mathbf{w}_{i-1})) > 0$, and $\Lambda(1, \mathbf{w}_{i+1}, \mathbf{w}_i, \mathbf{w}_{i-1}) < 0$.

Proof: Λ is shown to be decreasing by differentiating (17) with respect to κ and showing that $(\partial\Lambda/\partial\kappa) < 0$. The endpoint inequalities require application of the Lagrange remainder theorem. Details are given in [16] and [17]. ■

The next theorem shows that there exists a $\kappa \in [0,1]$ such that the path lengths are equal.

Theorem 7: If $\Lambda(\kappa, \mathbf{w}_{i-1}, \mathbf{w}_i, \mathbf{w}_{i+1})$ is given by (17), where $\beta \in [0,\pi)$ is given by (15) and $R = \hat{v}/c$, then there exists a unique $\kappa^* \in [0,1]$ such that $\Lambda(\kappa^*) = 0$. Furthermore, the κ-trajectory corresponding to κ^* has the same path length as the waypoint path $\overline{\mathbf{w}_{i-1}\mathbf{w}_i\mathbf{w}_{i+1}}$, i.e.

$$L = \|\mathbf{w}_i - \mathbf{w}_{i-1}\| + \|\mathbf{w}_{i+1} - \mathbf{w}_i\|.$$

In addition, a bisection search algorithm can be used to determine κ^* with accuracy $\bigcirc(2^{-n})$ where n is the number of function evaluations of Λ.

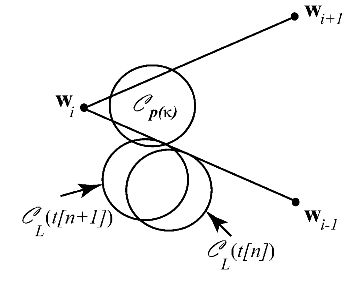

Fig. 6. Effect of fixed sample-rate on switching-time detection.

Proof: From Lemma 5 we know that Λ is the difference in path length between the κ-trajectory and the waypoint path. Therefore, if $\Lambda = 0$, the path lengths are equal. From Lemma 6, we know that there is a unique $\kappa^* =\in [0,1)$ such that $\Lambda(\kappa^*) = 0$.

Since $\Lambda(0) > 0$ and $\Lambda(1) < 0$ the first step of a bisection search is at $\kappa = 0.5$. The sign of $\Lambda(0.5)$ determines κ^* to within 2^{-1}. Subsequent function calls further refine the estimate. ■

V. Implementation Issues

In this section, we discuss several implementation issues that must be addressed to ensure robust behavior of the trajectory smoother when it is implemented in digital hardware. Real-time implementation requires that the differential (7)–(9) be solved via a numerical ODE solver (e.g., Runge–Kutta) using a fixed time-step. The fixed time-step creates several problems for detecting the switching times shown in Fig. 3. For example, suppose that the trajectory smoother is tracking the straight-line segment $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$ and the algorithm is looking for the intersection of $\mathcal{C}_L$ with $\mathcal{C}_{\mathbf{p}(\kappa)}$ in order to detect the switching time t_2, then we may have the scenario depicted in Fig. 6. The circle $\mathcal{C}_L$ may not intersect the circle $\mathcal{C}_{\mathbf{p}(\kappa)}$ exactly at the sample times. Therefore, we need a robust method for detecting circle and line intersections and for indicating when the intersection has been missed.

Toward that end, define the function

$$\begin{aligned}\Psi(\mathbf{a},\mathbf{b}) &\triangleq \text{sign}\left\{\left[\begin{pmatrix}\mathbf{a}\\0\end{pmatrix} \times \begin{pmatrix}\mathbf{b}\\0\end{pmatrix}\right]\cdot\hat{\mathbf{k}}\right\} \\ &= \text{sign}[a_x b_y - b_x a_y]\end{aligned} \tag{18}$$

where $\hat{\mathbf{k}}$ is the unit vector pointing into the plane. The function Ψ can be used for several purposes. First, if $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$ are unit vectors along the vectors $\overline{\mathbf{w}_{i-1}\mathbf{w}_i}$ and $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$, as shown in Fig. 2, then the direction of the turn is given by $\Psi(\mathbf{q}_i, \mathbf{q}_{i+1})$, where $\Psi(\mathbf{q}_i, \mathbf{q}_{i+1}) > 0$ indicates that the turn from the current path segment to the next path segment will be clockwise (a right-handed turn), and $\Psi(\mathbf{q}_i, \mathbf{q}_{i+1}) < 0$ indicates that the turn will be counterclockwise (a left-handed turn). The function Ψ can also be used to partition the $\mathbb{R}^2$ plane into two distinct halves, as shown in Fig. 7. Referring to Fig. 3, we will discuss the robust detection of switching times t_2 through t_5.

Switching Time t_2: Recalling that the center of circle $\mathcal{C}_L$ is $\mathbf{c}_L$ and that the center of circle $\mathcal{C}_{\mathbf{p}(\kappa)}$ is $\mathbf{p}(\kappa)$, it can be seen from Fig. 8 that the distance between the center of $\mathcal{C}_L$ and $\mathcal{C}_{\mathbf{p}(\kappa)}$ is given by $\|\mathbf{c}_L - \mathbf{p}(\kappa)\|$. Prior to switching time t_2, this distance is greater than $2R$; after switching time t_2, this distance is less than $2R$. Therefore the switching time t_2 can be determined by monitoring $\|\mathbf{c}_L - \mathbf{p}(\kappa)\|$, assuming that at some

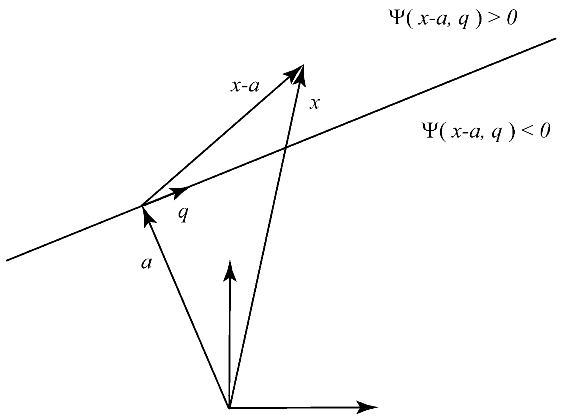

Fig. 7. Ψ divides $\mathbb{R}^2$ into two distinct subregions.

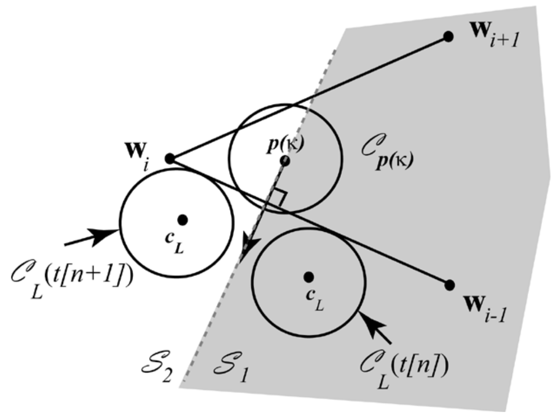

Fig. 8. Effect of finite sample-times for $\kappa \approx 1$.

sample time this quantity is less than $2R$. However, if $\kappa \approx 1$, then it is possible that $\|\mathbf{c}_L - \mathbf{p}(\kappa)\| > 2R$ at the sample times both before and after the intersection at time t_2, as shown in Fig. 8. Therefore, it is also necessary to detect when $\mathbf{c}_L$ has moved from the half plane $\mathcal{S}_1$ to the half plane $\mathcal{S}_2$. Guided by Fig. 7, this can be accomplished by monitoring the sign of $\Psi(\mathbf{c}_L - \mathbf{p}(\kappa), R(\pi/2)\mathbf{q}_i)$, where

$$R(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}.$$

If $\Psi(\mathbf{c}_L - \mathbf{p}(\kappa), R(\pi/2)\mathbf{q}_i) > 0$, then $\mathbf{c}_L \in \mathcal{S}_1$, otherwise $\mathbf{c}_L \in \mathcal{S}_2$.

Switching Time t_3: The switching time t_3 can also be detected via a half-plane argument. As shown in Fig. 9, t_3 occurs when the center of circle $\mathcal{C}_R(\mathbf{c}_R)$ transitions from $\mathcal{S}_1$ to $\mathcal{S}_2$, which can be detected with the function

$$\Psi\left(\mathbf{c}_R - \mathbf{c}_L, \frac{\mathbf{p}(\kappa) - \mathbf{c}_L}{\|\mathbf{p}(\kappa) - \mathbf{c}_L\|}\right).$$

Switching Time t_4: Robustly detecting switching time t_4 can be accomplished as follows. First, reflect the location of $\mathbf{c}_L$ at switching time t_3 through the line that bisects the angle β. The unit vector that defines the bisector of β is given by (12). The reflection of a vector through the line described by $\bar{\mathbf{q}}$ is given by the Householder transformation [18]

$$H(\bar{\mathbf{q}}) = \begin{pmatrix} 1 - 2\bar{q}_y^2 & 2\bar{q}_x\bar{q}_y \\ 2\bar{q}_x\bar{q}_y & 1 - 2\bar{q}_x^2 \end{pmatrix}.$$

Therefore, as shown in Fig. 10, we have $\mathbf{c}_L(t_4) = \mathbf{w}_i + H(\bar{\mathbf{q}})(\mathbf{c}_L(t_3) - \mathbf{w}_i)$.

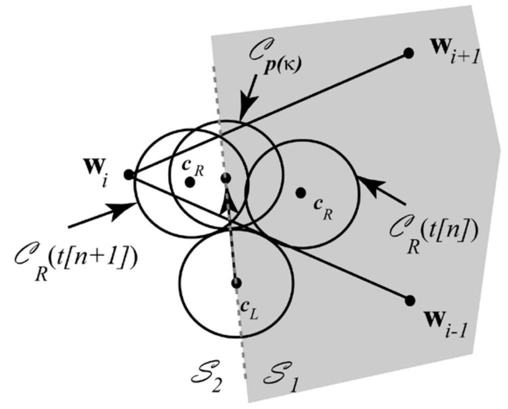

Fig. 9. Switching time t_3.

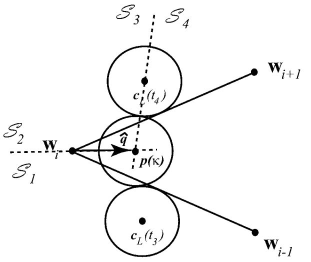

Fig. 10. Switching time t_4.

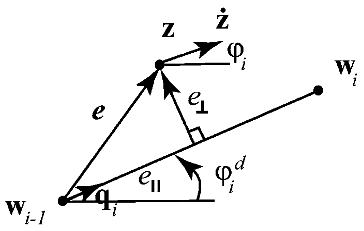

Fig. 11. Waypoint tracking.

The next step is to wait until $\mathbf{c}_L$ crosses into the half plane defined by $\bar{\mathbf{q}}$ that contains $\mathbf{c}_L(t_4)$. Referring to Fig. 10, we are looking for a transition from half plane $\mathcal{S}_1$ to $\mathcal{S}_2$. This is robustly identified when $\Psi(\mathbf{c}_L - \mathbf{w}_i, \bar{\mathbf{q}})$ switches from -1 to $+1$.

The last step is to look for the intersection of $\mathcal{C}_L$ with the line segment $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$. With reference to Fig. 10, this is robustly identified when $\mathbf{c}_L$ crosses from half plane $\mathcal{S}_3$ into half plane $\mathcal{S}_4$, or in other words, when

$$\Psi\left(\mathbf{c}_L - \mathbf{c}_L(t_4), \frac{\mathbf{p}(\kappa) - \mathbf{c}_L(t_4)}{\|\mathbf{p}(\kappa) - \mathbf{c}_L(t_4)\|}\right)$$

switches from -1 to $+1$.

Switching Time t_5: The detection of switching time t_5 is similar to the detection of switching time t_3.

Tracking: The final implementation issue is due to the fact that at switching time t_5, the trajectory generator, i.e., $\hat{\psi(0)}$, may not be aligned perfectly with the waypoint segment $\overline{\mathbf{w}_i\mathbf{w}_{i+1}}$; therefore setting $u = 0$ will cause it to drift from the waypoint segment. During the straight-line section a tracking algorithm is needed that causes the trajectory generator to asymptotically track the waypoint segment, while still satisfying the constraint $-c \leq u \leq c$.

Referring to Fig. 11, let $\psi_i^d = \tan^{-1}(q_{iy}/q_{ix})$ be the angle created by the waypoint path. To simplify the development, we

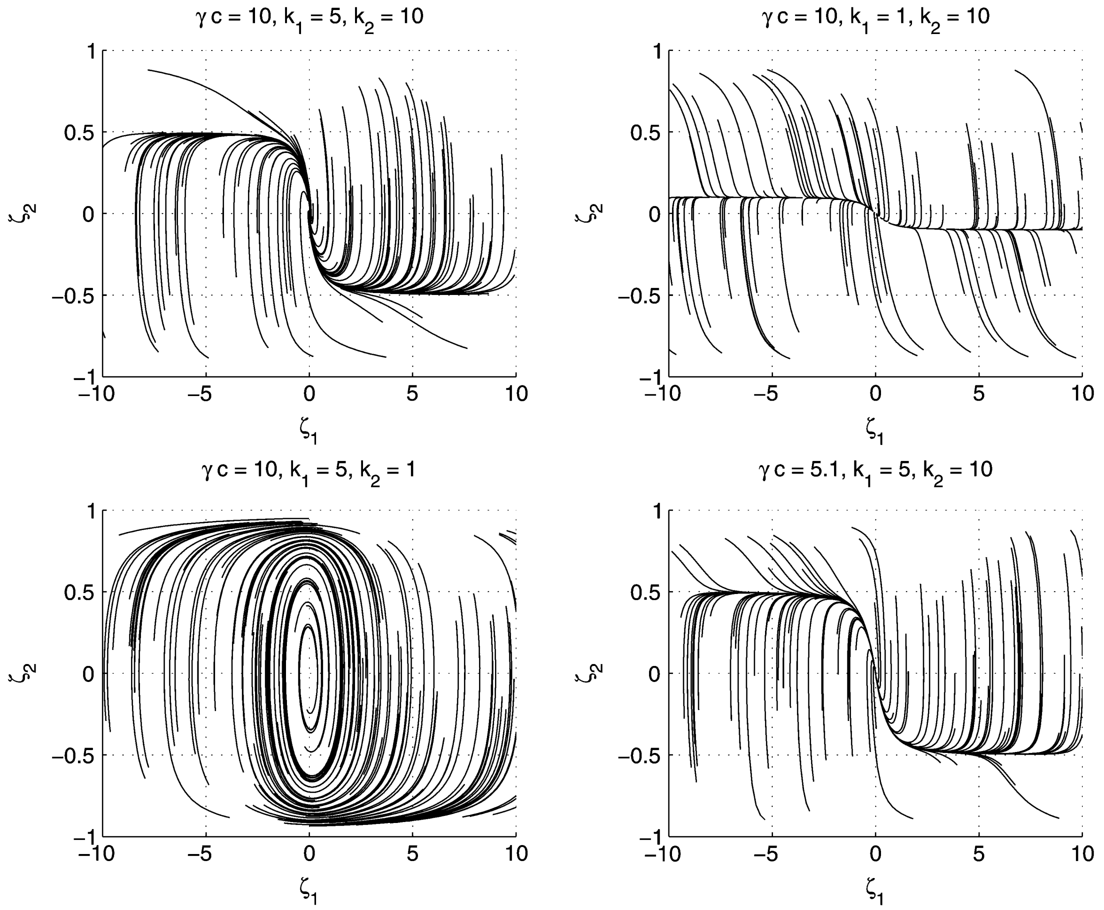

Fig. 12. Phase portrait for the transformed tracking problem.

will shift the origin to $\mathbf{w}_{i-1}$ and rotate the data by ψ_i^d. Accordingly we have

$$\mathbf{e} = \begin{pmatrix} e_{\|} \\ e_{\perp} \end{pmatrix} = R\left(\phi_i^d\right)(\mathbf{z} - \mathbf{w}_{i-1})$$
$$\dot{\mathbf{e}} = \begin{pmatrix} \dot{e}_{\|} \\ \dot{e}_{\perp} \end{pmatrix} = \begin{pmatrix} v\cos(\psi - \psi_i^d) \\ v\sin(\psi - \psi_i^d) \end{pmatrix}.$$

The control objective is to drive $e_\perp$ and $\psi - \psi_i^d$ to zero asymptotically, given the control constraint $u \in [-c, c]$. The following theorem describes how this can be done.

Theorem 8: Given the nonlinear system

$$\begin{aligned}\dot{e}_\perp &= v\sin(\psi - \psi_i^d)\\ \dot{\psi} &= u\end{aligned} \tag{19}$$

with control constraint $u \in [-c, c]$ where v is a positive constant. If

$$u = -\mathrm{sat}_c\left[\frac{\cos^3(\psi - \psi_i^d)}{\gamma}\left(\frac{k_1 e_\perp}{\sqrt{e_\perp^2 + v^2}} + k_2\sin(\psi - \psi_i^d)\right)\right] \tag{20}$$

where $\gamma c > 1, 0 < k_1 < \gamma c$, and $k_2 > 0$, then the origin is asymptotically stable and the domain of attraction is given by $\Omega = (-\infty, \infty) \times (\psi_i^d - \pi/2, \psi_i^d + \pi/2)$.

Proof: Using the change of variables

$$\begin{aligned}\zeta_1 &= e_\perp / v\\ \zeta_2 &= \sin(\psi - \psi_i^d)\end{aligned} \tag{21}$$

and the Lyapunov function

$$V(\zeta) = k_1\sqrt{\zeta_1^2 + 1} - 1 + \frac{\gamma}{2}\frac{\zeta_2^2}{1 - \zeta_2^2}$$

the proof follows by standard Lyapunov arguments [17]. ■

The phase portrait of system (19) with the control (20) after the transformation (21) is shown in Fig. 12 for several values of γ, k_1, and k_2. Fig. 12 shows the stability of the tracking error and the qualitative dependence of the transients on the parameters. Comparing the subplots on the left, it is seen that the damping characteristics are strongly dependent on k_2. Comparing the top subplots we see that the rise time of the tracking error is strongly dependent on k_1. It can also be seen that tracking response is not significantly influenced by γc.

VI. Simulation

The trajectory smoothing algorithm has a small computational load and can be run in real-time. In test runs of 200 000 iterations on a 1.8 GHz Pentium-class computer, the algorithm execution time exhibited a bimodal distribution. If the trajectory smoother was executing a turn, the average run time for one time-step was approximately 39 μs; if the trajectory smoother was executing a straight segment, the average run time for one step was approximately 16 μs. The computational simplicity of the algorithm enables its implementation in UAV applications where computational resources are modest.

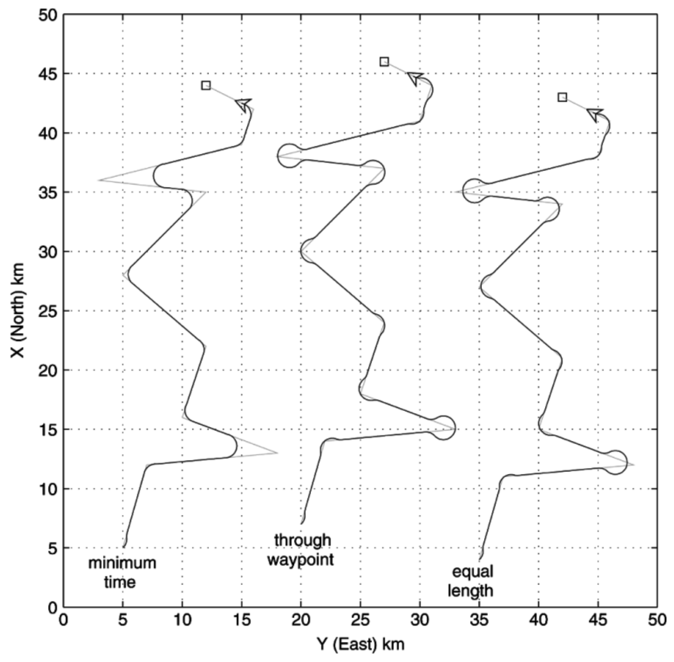

Fig. 13. Sample trajectories.

Simulation results for the trajectory smoother are shown in Fig. 13 for minimum-time transitions ($\kappa = 1$), transitions through the waypoint ($\kappa = 0$), and transitions matching the length of the original straight-line path (κ variable between 0 and 1). The smoothing algorithm was run in real time as the path was flown. As a turn on the path is completed, the smoothing algorithm looks to the next two waypoints and calculates the smooth transition for the next turn.

For the minimum-time transitions shown in Fig. 13, it can be seen that the location of the center of the turn circle corresponding to $\kappa = 1$ varies depending on the angle β between adjoining path segments. The same is true for the $\kappa = 0$ turn circles of the paths passing through the waypoints. For the equal-length path, the κ values for each turn that result in equal path lengths depend on the angle β between adjoining segments. For the last four turns of the equal-length path the κ values were 0.266, 0.136, 0.406, and 0.374. Typically, the more acute angles require smaller κ values to equalize the length.

VII. Conclusion

A method for generating time-extremal trajectories for transitioning between successive waypoint path segments has been developed. These paths satisfy kinematic input constraints that model the dynamic capabilities of a UAV and have been implemented via a simple, real-time algorithm. In addition, a method for generating trajectories with the same length as the original straight-line path has been developed.

There are several advantages to the dynamic trajectory smoothing approach. First, it integrates easily with waypoint path planning algorithms that produce straight-line paths. Second, the approach has low computational overhead. In fact, trajectories are generated in real-time, as the vehicle transitions along the path. Third, the dynamic trajectory smoother minimizes the time that the vehicle deviates from the straight-line path. These advantages make this approach a viable alternative for implementation in UAV applications.

References

[1] M. Pachter and P. R. Chandler, "Challenges of autonomous control," *IEEE Control Syst. Mag.*, vol. 18, no. 4, pp. 92–97, Aug. 1998.
[2] P. R. Chandler, S. Rasumussen, and M. Pachter, "UAV cooperative path planning," in *Proc. AIAA Guidance, Navigation, Control Conf.*, AIAA Paper AIAA-2000-4370, Denver, CO, Aug. 2000.
[3] T. McLain and R. Beard, "Cooperative rendezvous of multiple unmanned air vehicles," in *Proc. AIAA Guidance, Navigation Control Conf.*, Paper AIAA-2000-4369, Denver, CO, Aug. 2000.
[4] R. W. Beard, T. W. McLain, M. Goodrich, and E. P. Anderson, "Coordinated target assignment and intercept for unmanned air vehicles," *IEEE Trans. Robot. Autom.*, vol. 18, no. 6, pp. 911–922, Dec. 2002.
[5] P. R. Chandler, M. Pachter, D. Swaroop, J. M. Fowler, J. K. Howlett, S. Rasmussen, C. Schumacher, and K. Nygard, "Complexity in UAV cooperative control," in *Proc. Amer. Control Conf.*, Anchorage, AK, May 2002, pp. 1831–1836.
[6] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *Amer. J. Math.*, vol. 79, pp. 497–516, 1957.
[7] G. Yang and V. Kapila, "Optimal path planning for unmanned air vehicles with kinematic and tactical constraints," in *Proc. IEEE Conf. Decision Control*, Las Vegas, NV, 2002, pp. 1301–1306.
[8] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *J. Guid. Control Dyn.*, vol. 25, no. 1, pp. 116–129, Jan.–Feb. 2002.
[9] N. Faiz, S. K. Agrawal, and R. M. Murray, "Trajectory planning of differentially flat systems with dynamics and inequalities," *J. Guid. Control Dyn.*, vol. 24, no. 2, pp. 219–227, Mar.–Apr. 2001.
[10] R. K. Prasanth, J. D. Boskovic, S.-M. Li, and R. K. Mehra, "Initial study of autonomous trajectory generation for unmanned aerial vehicles," in *Proc. IEEE Conf. Decision Control*, Orlando, FL, Dec. 2001, pp. 640–645.
[11] O. A. Yakimenko, "Direct method for rapid prototyping of near-optimal aircraft trajectories," *J. Guid. Control Dyn.*, vol. 23, no. 5, pp. 865–875, Sep.–Oct. 2000.
[12] S. Sun, M. B. Egerstedt, and C. F. Martin, "Control theoretic smoothing splines," *IEEE Trans. Autom. Control*, vol. 45, no. 12, pp. 2271–2279, Dec. 2000.
[13] A. W. Proud, M. Pachter, and J. J. D'Azzo, "Close formation flight control," in *Proc. AIAA Guidance, Navigation, Control Conf. Exhibit*, Paper AIAA-99-4207, Portland, OR, Aug. 1999, pp. 1231–1246.
[14] R. L. Burden and J. D. Faires, *Numerical Analysis*, 4th ed. Boston, MA: PWS-Kent, 1988.
[15] T. L. Vincent and W. J. Grantham, *Nonlinear and Optimal Control Systems*. New York: Wiley, 1997.
[16] E. P. Anderson, "Constrained Extremal trajectories and unmanned air vehicle trajectory generation," M.S. thesis, Brigham Young Univ., Provo, UT, Apr. 2002.
[17] E. P. Anderson, R. W. Beard, and T. W. McLain. (2004, Apr.) Dynamic trajectory smoothing for UAVs. Brigham Young Univ., Provo, UT. [Online]. Available: http://hdl.handle.net/1877/22
[18] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms*. Englewood Cliffs, NJ: Prentice-Hall, 2000.