2003-02-01

Real Time Dynamic Trajectory Smoothing for Uninhabited Aerial Vehicles

Erik Anderson

Randal Beard
beard@byu.edu

See next page for additional authors

Follow this and additional works at: https://scholarsarchive.byu.edu/facpub

Part of the Electrical and Computer Engineering Commons

Original Publication Citation

BYU ScholarsArchive Citation
Anderson, Erik; Beard, Randal; and McLain, Timothy W., "Real Time Dynamic Trajectory Smoothing for Uninhabited Aerial Vehicles" (2003). All Faculty Publications. $15.
https://scholarsarchive.byu.edu/facpub/515

This Peer-Reviewed Article is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Faculty Publications by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
Real Time Dynamic Trajectory Smoothing for Uninhabited Aerial Vehicles

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

E. P. Anderson is a graduate research assistant in the Department of Electrical Engineering at Stanford University, Stanford, CA 94305. R. W. Beard is an associate professor in the Electrical and Computer Engineering Department at Brigham Young University, Provo, Utah 84602. T. W. McLain is an associate professor in the Mechanical Engineering Department at Brigham Young University, Provo, Utah 84602. Email: eandersn@stanford.edu, beard@ee.byu.edu, tmclain@et.byu.edu. R.W. Beard is the corresponding author.

February 7, 2003
Abstract

This paper presents a real-time, feasible, trajectory generation algorithm for unmanned air vehicles (UAVs) flying through a sequence of waypoints. Pontryagin's Minimum Principle is used to demonstrate that the transitions generated by the algorithm between straight-line path segments connecting waypoints are time optimal. 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.

Keywords

Trajectory Generation, Uninhabited Aerial Vehicles, optimal control.

I. Introduction

Recent advances in computing, sensing, and battery technology have made unmanned vehicle technology a viable option in both military and commercial applications[1], [2]. The grand vision of the U.S. armed forces is that by the year 2020, thirty percent of its forces will be unmanned[3]. In order to make unmanned vehicle technology more useful, it is critical that autonomous, semi-autonomous, and cooperative behaviors be developed. Central to all of these behaviors are automatic trajectory generation algorithms.

For uninhabited aerial vehicles (UAVs), the trajectory generation problem is complicated by the following considerations. First, UAVs are nonlinear dynamic systems of relatively high order. The nonlinear dynamics of the vehicle pose several constraints on feasible trajectories. In particular, the heading rate constraint of the UAV places curvature constraints on the trajectories, while the stall and aerodynamic thrust limitations result in constraints on the minimum and maximum velocity of the vehicle. Therefore, the UAV trajectory generation problem is inherently different than the trajectory generation problem for mobile robots, which do not have velocity or heading rate constraints. A second challenge is that the trajectory generator must operate in real-time to be responsive to changing operational situations, and to enable autonomy and cooperation.

Our particular interest has been the cooperative timing problem for UAVs[4], [5], [6]. For example, in the simultaneous intercept problem, teams of UAVs must maneuver through a dynamically changing threat field to arrive at their individual destinations simultaneously with the other UAVs. As threats pop-up, the entire team must re-plan their paths to satisfy the simultaneous arrival constraint. To solve this problem, we have recently introduced a cooperative control technique [7], [8] that is based upon the hierarchical architecture shown in Figure 1. As shown in Figure 1, the Coordinated Timing Manager produces a set of targets with associated Estimated-Time-of-Arrival (ETA) data. Our approach to the Coordinated Timing Manager is detailed in [7], [8]. Of particular importance to this paper is the fact that the Coordinated Timing Manager plans cooperative tasks based upon straight-line waypoint paths. The role of the Dynamic Trajectory Smoother (DTS) is to smooth the waypoint paths into time-parameterized trajectories that satisfy the dynamic constraints of the UAV. However, if the DTS produces a trajectory with a path length that is different than the original waypoint path, then the estimated time-of-arrival will
be changed, thus invalidating the cooperative timing plan produced by the Cooperative Timing Manager. Therefore, we would like the DTS to produce trajectories with the same path length as the original waypoint path.

These considerations pose an interesting challenge: design a trajectory generation algorithm that
- Smooths through a set of waypoints, minimizing the deviation from the associated waypoint path,
- Satisfies the curvature and velocity constraints imposed by the dynamics of the UAV,
- Maintains the path length of the associated waypoint path, and
- Operates in real-time.

The purpose of this paper is to introduce a trajectory generation algorithm that meets these objectives.

While the trajectory generation technique introduced in this paper is ideally 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 is carrying a sensor with a finite dimensional footprint[9]. 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. Therefore, we will expand the problem statement to include real-time feasible trajectories that can be configured to either (1) produce trajectories with the same path length as the original waypoint path, (2) pass directly through the waypoint while minimizing the deviation from the original waypoint path, (3) transition between the waypoints in minimum time, or (4) pass a specified distance from the
waypoint, while minimizing the deviation from the straight-line segments.

Our approach is based on the local reachability region of the aircraft and basic geometry. In that sense, it is similar to the approaches reported in [4], [10], [11]. In [10] it is shown that the shortest path between two points satisfying curvature constraints, is comprised of circles and straight-line path segments. Reference [11], builds upon Dubins 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 [4], Dubins circles are superimposed as fillets at the junction of straight-line waypoint paths produced from a Voronoi diagram. Our approach differs from [4], [10], [11] 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 the algorithm temporally distributes the computations, making the algorithm feasible to implement in real-time.

Several other approaches to UAV path planning have recently appeared in the literature. The approaches can be roughly grouped into three categories: probabilistic maps, differential flatness, and optimal control techniques.

The probabilistic roadmap approach, which is described in [12] uses a set of motion primitives for each vehicle to randomly generate a set of path segments. If the generated path segment is feasible (e.g., collision free), then that segment is added to a tree of possible paths. The algorithm continues to expand the tree until the goal configuration is reached. The application of probabilistic roadmaps to unmanned helicopters is described in [13], [14]. Unfortunately this work is not directly applicable to the current problem since it relies upon the hovering capability of the helicopter. Reference [15] describes analysis techniques for probabilistic roadmaps. The disadvantage of probabilistic roadmaps is that they are only guaranteed to find feasible paths with probability one when computation time is unbounded.

Differential flatness techniques have been applied recently to the UAV trajectory generation problem. An excellent overview of differential flatness with application to mobile robots is given in [16]. A path planning/trajectory generation approach for mobile robots using differential flatness is described in [17]. A computational approach to generating feasible UAV trajectories based on differential flatness is given in [18]. An extension to [18] which is based on LMIs is given in [19]. In Reference [20], a notion of group flatness is introduced and applied to the problem UAV formation flying. The disadvantage with differential flatness approaches is that it is often difficult to represent coordination constraints as convex regions of the flat space. In addition, the optimization techniques required to find trajectories in the flat space are often infeasible in real-time.

Optimal control approaches to path planning attempt to minimize a cost index subject to the feasibility constraints of the vehicle. In [21], trajectories are parameterized as splines and nonlinear optimization techniques are used to find the spline coefficients. Reference [22] uses splines and nonlinear optimization techniques to smooth through straight-line waypoint paths. Reference [23] derives an interesting method
for creating splines trajectories for waypoint paths that do not pass through the waypoints but rather
smoothes through the points. In Reference [24], Pontryagin’s maximum principle is used to show that
optimal paths for mobile robots consist of a series of rotations and straight line maneuvers. Pontryagin’s
principle is used in [25], to show that if the robot’s velocity is constant, then optimal trajectories consist
of sequences of constant velocity arcs and straight line segments. The difficulty with optimal control
techniques is that they are only computationally feasible for simple constraints and configurations.

The main contribution of this paper is the development of a trajectory generation technique that
smoothes waypoint paths into dynamically feasible trajectories. The salient features of our algorithm
are that it has low computational complexity, thus facilitating real-time implementation, it is reactive to
dynamic environments, and it is easily configured to satisfy path-length constraints, such as equal path
length, or passing within a specified distance of a waypoint.

The remainder of the paper will be organized as follows. Problem definition and mathematical prelimi-
naries will be given in Section II. In Section III we define a class of trajectories which we call \( \kappa \)-trajectories,
and show that \( \kappa \)-trajectories are constrained time-optimal trajectories in the sense that they satisfy Pon-
tryagin’s minimum principle [26]. The central result of the paper is presented in Section IV, where we
introduce a real-time algorithm that generates \( \kappa \)-trajectories, and show how it can be configured to gen-
erate, minimum-transition-time trajectories, waypoint trajectories, and trajectories with the same path
length as the original waypoint path. In Section V we discuss several implementation issues required to
make the algorithm work on digital hardware. Section VI demonstrates the effectiveness of the algorithm
in several simulated scenarios. Finally Section VII offers some conclusions.

II. PROBLEM STATEMENT AND PRELIMINARIES

In defining the problem, we will assume that the UAV is flying at constant altitude and is equipped with
an inner-loop autopilot capable of receiving velocity and heading commands. Following [27] we assume
that the UAV dynamics subject to velocity-hold and heading-hold autopilots are first order:

\[
\begin{align*}
\dot{x} &= v \cos \psi \\
\dot{y} &= v \sin \psi \\
\dot{\psi} &= \alpha_{\psi} (\psi^c - \psi) \\
\dot{v} &= \alpha_{v} (v^c - v)
\end{align*}
\]

where \( \alpha_{\psi} \) and \( \alpha_{v} \) 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:

\[
\begin{align*}
-c &\leq \dot{\psi} \leq c \\
0 &< v_{\text{min}} \leq v \leq v_{\text{max}}
\end{align*}
\]
Definition 1: A trajectory $z^d(t) = (z^d_x, z^d_y)^T$ is called dynamically feasible if there exist inputs $\psi^c(t)$ and $v^c(t)$ such that $z(0) = z^d(0)$ implies that $z(t) = z^d(t)$, and the dynamics (1)–(4) and constraints (5)–(6) are satisfied, for all $t \geq 0$.

The input to the Dynamic Trajectory Smoother (DTS) shown in Figure 1 is a waypoint path

$$\mathcal{P} = \{v, \{w_1, w_2, \ldots, w_N\}\},$$

where $v \in [v_{\text{min}}, v_{\text{max}}]$ is the desired velocity of the UAV, and $w_i \in \mathbb{R}^2$ denote the waypoints expressed in inertial coordinates.

The essential idea of our approach is to give the trajectory generator a mathematical structure that resembles the UAV dynamics. In particular the DTS is given by the differential equations

$$\begin{align*}
\dot{\hat{z}}_x &= \hat{v} \cos \hat{\psi} \\
\dot{\hat{z}}_y &= \hat{v} \sin \hat{\psi} \\
\dot{\hat{\psi}} &= u
\end{align*}$$

where $u \in [-c, c]$ is an input that will be selected to meet the specified objectives. We will assume that the trajectory is to be traversed at a constant velocity $\hat{v} \in [v_{\text{min}}, v_{\text{max}}]$. Note that if $\hat{z}(0) = z(0)$, then Equations (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 ODE solver and propagated in real-time. In other words, the output of the DTS corresponds in time to the evolution of the UAV dynamics. If a forth-order Runge-Kutta algorithm [28] is used then $u$ will need to be computed four times each sample period. Therefore, the computational complexity depends upon the computation of $u$.

Note that if $u = +c$, then the DTS given in Equations (7)–(8) traces out a right-handed circle as shown in Figure 2. Similarly, if $u = -c$ then the DTS traces out a left-handed circle. As shown in Figure 2, the local reachability region of the DTS is bounded by these two circles. The radius of the circles defining the local reachability region is given by $R = \hat{v}/c$. Note that as the desired velocity increases, the minimum turning radius increases. If $(\hat{z}, \hat{\psi})$, where $\hat{z} = (\hat{z}_x, \hat{z}_y)^T$, is the configuration of the DTS, then the centers of the two circles that bound the reachability region are given by

$$\begin{align*}
\mathbf{c}_R(t) &= \hat{z}(t) + R \begin{pmatrix}
-\sin(\hat{\psi}(t)) \\
\cos(\hat{\psi}(t))
\end{pmatrix} \\
\mathbf{c}_L(t) &= \hat{z} + R \begin{pmatrix}
\sin(\hat{\psi}(t)) \\
-\cos(\hat{\psi}(t))
\end{pmatrix}
\end{align*}$$

The corresponding circles of radius $R$ are denoted by $C_L$ and $C_R$. 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.
In Section III, we will define a class of feasible trajectories and show that they satisfy Pontryagin’s necessary conditions for minimizing the transition time from one waypoint segment to the next, subject to the dynamics (1)–(4) and constraints (5)–(6). In order to be precise we recall Pontryagin’s minimum principle, [26] which can be stated as follows.

**Definition 2:** Given the dynamical system \( \dot{x} = f(x,u) \) with initial condition \( x_0 \), the terminal constraint set \( \chi = \{ x \in \mathbb{R}^n : g(x) = 0 \} \) and the control constraint set \( U = \{ u \in \mathbb{R}^m : h(u) \geq 0 \} \) where \( f, g : \mathbb{R}^n \rightarrow \mathbb{R}^q \), and \( h \) are continuously differentiable and \( \frac{\partial g}{\partial x} \) and \( \frac{\partial h}{\partial u} \) are full rank, the control strategy \( u^*(t) \) is an extremal control strategy with respect to the cost function \( J = \int_0^T l(x,u) dt \) where \( l \) is continuously differentiable, if there exists a continuous piecewise differentiable function \( \lambda(t) \in \mathbb{R}^n \) and a constant \( \rho \in \mathbb{R}^q \) such that the following equations are satisfied

\[
\begin{align*}
\dot{x}^* &= \frac{\partial H}{\partial \lambda}(x^*,u^*,\lambda^*) \\
\dot{\lambda}^* &= -\frac{\partial H}{\partial x}(x^*,u^*,\lambda^*) \\
\lambda^*(T) &= \frac{\partial g}{\partial x}(x^*(T))^T \rho \\
g(x^*(T)) &= 0 \\
u^*(t) &= \arg \min_{u \in U} H(x^*,u,\lambda^*) \\
H(x^*,u^*,\lambda^*) &= 0,
\end{align*}
\]

where \( H(x,u,\lambda) = l(x,u) + \lambda^T f(x,u) \) is the Hamiltonian of the system.

A minimum-time-extremal trajectory is an extremal trajectory with respect to the cost function \( J = \int_0^T dt \).
III. Extremal Trajectories

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

Consider the waypoint path defined by the three waypoints $w_{i-1}$, $w_i$, and $w_{i+1}$, and let

$$q_i = \frac{w_i - w_{i-1}}{\|w_i - w_{i-1}\|}$$
$$q_{i+1} = \frac{w_{i+1} - w_i}{\|w_{i+1} - w_i\|}$$

denote unit vectors along the corresponding path segments as shown in Figure 3. Letting $\beta$ denote the angle between $q_i$ and $q_{i+1}$ we get

$$\beta = \cos^{-1} \left( q_{i+1}^T q_i \right).$$

As shown in Figure 3, let $\tilde{C}$ 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 $\overline{w_{i-1}w_i}$ and $\overline{w_iw_{i+1}}$ at exactly one point each. The bisector of $\beta$ will intersect $\tilde{C}$ at two locations. Let $\tilde{p}$ be the intersection that is closest to $w_i$.

From Figure 3, it can be seen that

$$\sin \left( \frac{\beta}{2} \right) = \frac{R}{\|\tilde{p} - w_i\| + R}.$$ 

Manipulating this expression, the distance between $w_i$ and $\tilde{p}$ is found to be

$$\|\tilde{p} - w_i\| = R \left( \frac{1}{\sin \left( \frac{\beta}{2} \right)} - 1 \right).$$

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

$$\tilde{q} = \frac{q_{i+1} - q_i}{\|q_{i+1} - q_i\|}. \quad (18)$$
The point $\bar{p}$ is therefore given by

$$\bar{p} = w_i + R \left( \frac{1}{\sin \left( \frac{\beta}{2} \right)} - 1 \right) \bar{q}.$$  

Let $p(\kappa)$ denote a parameterized point on the line between $w_i$ and $\bar{p}$, on the bisector of the angle:

$$p(\kappa) = w_i + \kappa R \left( \frac{1}{\sin \left( \frac{\beta}{2} \right)} - 1 \right) \bar{q}, \quad (19)$$

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

As shown in Figure 4, let $C_{p(\kappa)}$ be a circle of radius $R = \hat{v}/c$ whose center lies in the direction of $\bar{q}$ and intersects $p(\kappa)$. Also, let $C_i$ be a circle of radius $R$ placed such that it intersects $C_{p(\kappa)}$ and $w_i w_{i-1}$ in exactly one location each. Define $C_{i+1}$ similarly, as shown in Figure 4.

**Definition 3:** A $\kappa$-trajectory is defined as the trajectory that is constructed by following the line segment $w_i w_{i+1}$ until intersecting $C_i$, which is followed until $C_{p(\kappa)}$ is intersected, which is followed until intersecting $C_{i+1}$ which is followed until the line segment $w_i w_{i+1}$ is intersected, as shown in Figure 4.

![Fig. 4. A dynamically feasible $\kappa$-trajectory.](image-url)

**Theorem 4:** The $\kappa$-trajectory shown in Figure 4 is the unique, dynamically feasible, minimum-time-extremal trajectory that transitions from the waypoint segment $w_i w_{i-1}$ to the waypoint segment $w_i w_{i+1}$ and passes directly through $p(\kappa)$.

**Proof:** Note that the $\kappa$-trajectory shown in Figure 4 is symmetric about the bisector of $\beta$. Due to symmetry it is sufficient to show that the trajectory from $p(\kappa)$ to the line segment $w_i w_{i+1}$ is minimum-time extremal. Without loss of generality, perform a change of coordinates such that the line segment $w_i w_{i+1}$ lies along the Y-axis as shown in Figure 5. Consider the time-optimal control problem with initial
conditions

\[
\begin{pmatrix}
\dot{z}(0) \\
\dot{\psi}(0)
\end{pmatrix} = \begin{pmatrix}
p(\kappa) \\
\theta \equiv \frac{\pi - \beta}{2}
\end{pmatrix},
\]

and terminal constraints given by

\[
g(x) = \begin{pmatrix}
n^T \dot{z}(T) \\
\dot{\psi}(T)
\end{pmatrix} = 0,
\]

where \( n = (1, 0)^T \) is the normal vector to the line segment \( \overline{w_iw_{i+1}} \), and \( \dot{\psi}(T) = 0 \). The terminal constraint ensures that at time \( T \), the DTS is on the line segment and is aligned in the desired direction. The Hamiltonian for the system is given by

\[
H = 1 + \lambda_1 \dot{v} \cos \dot{\psi} + \lambda_2 \dot{v} \sin \dot{\psi} + \lambda_3 u.
\]

We conjecture that the bang-bang control strategy

\[
u^*(t) = \begin{cases} 
-c, & 0 \leq t < t_1 \\
c, & t_1 \leq t \leq T
\end{cases}
\]

is minimum-time extremal and show that the resulting system trajectories satisfy Equations (12)–(17). Equation (12) is trivially satisfied since \( \partial H/\partial \lambda = (\dot{v} \cos \dot{\psi}, \dot{v} \sin \dot{\psi}, u)^T \). Equations (13) and (14) results in
system of ODEs
\[
\begin{align*}
\dot{\lambda}_1^* &= 0 \\
\dot{\lambda}_2^* &= 0 \\
\dot{\lambda}_3^* &= \lambda_1^* \hat{v} \sin \hat{\psi}^* - \lambda_2^* \hat{v} \cos \hat{\psi}^*
\end{align*}
\]
with terminal constraints
\[
\begin{align*}
\lambda_1^*(T) &= \rho_1 n_x = 0, \\
\lambda_2^*(T) &= \rho_1 n_y = \rho_1, \\
\lambda_3^*(T) &= \rho_2.
\end{align*}
\]
Therefore \(\lambda_1^*(t) \equiv 0\) and \(\lambda_2^*(t) \equiv \rho_1\). Integrating \(\dot{\hat{\psi}}^* = u^*\) backward in time from \(t = T\) and forward in time from \(t = 0\) gives
\[
\hat{\psi}^*(t) = \begin{cases} 
-ct + \theta, & 0 \leq t \leq t_1 \\
(c(t - T)), & t_1 \leq t \leq T.
\end{cases}
\tag{21}
\]
Enforcing continuity at \(t = t_1\) gives
\[t_1 = \frac{T}{2} + \frac{\theta}{2c}.\]
Integrating \(\dot{\lambda}_3\) backward in time from \(t = T\) gives
\[
\lambda_3^*(t) = \begin{cases} 
\frac{\rho_1 \hat{v}}{c} \sin(-ct + \theta) + k_1, & 0 \leq t \leq t_1 \\
-\frac{\rho_1 \hat{v}}{c} \sin(c(t - T)) + k_2, & t_1 \leq t \leq T,
\end{cases}
\tag{22}
\]
where
\[
k_1 = -\frac{2\rho_1 \hat{v}}{c} \sin \left(\frac{\theta}{2} - \frac{cT}{2}\right) + \rho_2
\]
\[
k_2 = \rho_2
\]
are determined from continuity and terminal conditions. Evaluating the Hamiltonian along \((x^*, \lambda^*, u^*)\) we get
\[
H(x^*, \lambda^*, u^*) = 1 + \lambda_2^* \hat{v} \sin(\hat{\psi}^*) + \lambda_3^* u^* = \begin{cases} 
1 - ck_1, & 0 \leq t \leq t_1 \\
1 + ck_2, & t_1 \leq t \leq T.
\end{cases}
\]
Therefore Eq. (17) requires that \(\rho_2 = -1/c\) and
\[
\rho_1 = -\frac{1}{\hat{v} \sin \left(\frac{\theta}{2} - \frac{cT}{2}\right)},
\tag{23}
\]
which is finite provided that \(cT - \theta \neq 2\pi n\) for any integer \(n\). By finding \(T\) explicitly, we will show that \(\rho_1\) is well defined. Physically, \(cT - \theta\) can only equal \(2\pi\) (or integer multiples) if looping occurs in the transition from one segment to the next.
Integrating $\dot{\hat{z}}_y$ we obtain

$$\dot{\hat{z}}_y = \begin{cases} \frac{\dot{\hat{v}}}{c} \cos(-ct + \theta) - \frac{\dot{\hat{v}}}{c} \cos \theta, & 0 \leq t \leq t_1 \\ -\frac{\dot{\hat{v}}}{c} \cos(c(t - T)) + \frac{2\dot{\hat{v}}}{c} \cos \left( \frac{\theta}{2} - \frac{T}{2} \right) - \frac{\dot{\hat{v}}}{c} \cos \theta, & t_1 \leq t \leq T \end{cases}. \quad (24)$$

We choose $T$ to be the smallest positive number such that $\dot{\hat{z}}_y(T) = 0$. From Eq. (24), $\dot{\hat{z}}_y(T) = 0$ implies that

$$\cos \left( \frac{\theta}{2} - \frac{cT}{2} \right) = \frac{1 + \cos \theta}{2}. \quad (25)$$

Since $0 < \frac{1 + \cos \theta}{2} < 1$, there always exists a real $\theta/2 - cT/2$ that satisfies Eq. (25). Hence

$$T = \frac{\theta}{c} - \frac{2}{c} \cos^{-1} \left( \frac{1 + \cos \theta}{2} \right). \quad (26)$$

Define

$$\Gamma(\alpha) \triangleq \frac{\theta}{c} - \frac{2}{c} \alpha. \quad (27)$$

We now determine which angle is to be chosen for the $\arccos(\cdot)$ expression in (26). $\alpha \equiv \arccos(\beta)$ where $0 < \beta < 1$ has one solution $\alpha_1$ where $0 < \alpha_1 < \frac{\pi}{2}$ and another solution $\alpha_2$ where $-\frac{\pi}{2} < \alpha_2 < 0$. In fact, $\alpha_2 = -\alpha_1$. If we choose $\alpha$ such that $\alpha > \frac{\pi}{2}$ then $T$ will be negative and an unacceptable solution results.

On the other hand, if we choose $\alpha$ such that $\alpha < -\frac{\pi}{2}$, i.e. $\alpha = \alpha_2 - 2\pi n = -\alpha_1 - 2\pi n$ where $n$ is a positive integer, then $0 < \Gamma(\alpha_2) < \Gamma(\alpha)$. Hence in our search for the smallest positive $T$, we need only consider $\alpha = \alpha_1$ or $\alpha_2$. Therefore, the smallest positive $T$ is given by

$$T = \min_{\Gamma(\alpha) \geq 0} \{ \Gamma(\alpha_1), \Gamma(\alpha_2) \}. \quad (28)$$

Returning to the question as to whether $\rho_1$ as expressed in Eq. (23) is well-defined, note that by (25) and since $0 < \theta < \frac{\pi}{2}$, we have $0 < \frac{1 + \cos \theta}{2} < 1$. Hence $\theta/2 - cT/2 \neq n\pi$ for any integer $n$ and the equation for $\rho_1$ is well-defined. We also see that $\rho_1$ is nonzero, so $\lambda_2^*(t) \neq 0$.

We have shown that Conditions (12)-(17) are satisfied. For completeness we also give the expression for $\dot{\hat{z}}_x$:

$$\dot{\hat{z}}_x(t) = \begin{cases} -\frac{\dot{\hat{v}}}{c} \sin(-ct + \theta) + \frac{\dot{\hat{v}}}{c} \sin \theta & 0 \leq t \leq t_1 \\ \frac{\dot{\hat{v}}}{c} \sin(c(t - T)) - \frac{2\dot{\hat{v}}}{c} \sin \left( \frac{\theta}{2} - \frac{T}{2} \right) + \frac{\dot{\hat{v}}}{c} \sin \theta & t_1 \leq t \leq T \end{cases}. \quad (29)$$

IV. Dynamic Trajectory Smoothing

In this section we introduce a real-time algorithm that generates $\kappa$-trajectories. If $\kappa = 1$, then the $\kappa$-trajectory transitions from the waypoint segment $w_{i-1}w_i$ to the waypoint segment $w_iw_{i+1}$ in minimum time. If $\kappa = 0$, then the $\kappa$-trajectory executes a minimum-time transition subject to the constraint that it pass directly over $w_i$. We will also show in this section how to select $\kappa$ so that the $\kappa$-trajectory has the same path length as the original waypoint path.
The essential idea for the algorithm is shown in Figure 6. The right and left turning constraints given in Eqs. (10)–(11) are denoted by $C_R$ and $C_L$ respectively at different time instances. The progression of time is denoted by $t_1, \ldots, t_6$. The right turning constraint $C_R$ is not shown at times $t_2$ and $t_5$ to avoid cluttering the figure. At time $t_1$ the DTS is tracking the waypoint segment $\mathbf{w}_{i-1}\mathbf{w}_i$. When the left turning circle $C_L$ intersects $C_p(\kappa)$ at time $t_2$, $u$ is set to $-c$. The left turning constraint is followed until the right turning circle $C_R$ corresponds exactly with $C_p(\kappa)$ at time $t_3$. The DTS variable $u$ is then set to $+c$ and the right turning constraint is followed until the left turning constraint $C_L$ intersects the waypoint segment $\mathbf{w}_i\mathbf{w}_{i+1}$ at time $t_4$. The DTS variable $u$ 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 DTS selection scheme for $u$ is shown in Figure 7. The nominal state of the DTS 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 $u = 0$. The tracking algorithm that is used will be discussed in Section V. When the DTS begins tracking the current waypoint segment, the location of $C_p(\kappa)$ is computed. If the turn is a clockwise turn then the constraint circle $C_L$ is monitored until it intersects $C_p(\kappa)$ at which point $u \leftarrow -c$ ($t_2$ in Figure 6). When $u = -c$ the motion of the DTS is such that $C_L$ is stationary. The constraint circle $C_R$ is monitored until it coincides exactly with $C_p(\kappa)$ at which time $u \leftarrow +c$ (time $t_3$). $C_R$ is now stationary and $C_L$ is monitored until it intersects the line segment $\mathbf{w}_i\mathbf{w}_{i+1}$ from the right, at which time $u \leftarrow -c$ (time $t_4$). The constraint circle $C_L$ is stationary and $C_R$ is monitored until it no longer intersects $\mathbf{w}_i\mathbf{w}_{i+1}$, at which point 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. There are practical issues associated with finding these switching times in digital hardware. These issues and the associated implications will be discussed in Section V.
If $\kappa = 1$, then the DTS algorithm simplifies considerably. Similar to the case when $\kappa \in [0,1)$, the first step is to determine $C_{p(\kappa)}$ and the direction of the turn. As shown in Figure 8, for a clockwise turn the DTS tracks the straight-line path segment $w_{i-1}w_i$ until $C_R$ coincides with $C_{p(\kappa)}$, at which point $u \leftarrow +c$ ($t_2$ in Figure 8). $C_R$ is then stationary and $C_L$ is monitored until the entire circle is to the left of $w_iw_{i+1}$ (time $t_3$), at which time tracking is resumed.

The following theorem asserts that the DTS algorithm implements the extremal $\kappa$-trajectories defined in Section III.

**Theorem 5**: The DTS algorithm depicted in Figure 7, implements the extremal $\kappa$-trajectories defined
in Section III.

The DTS algorithm 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 $\kappa$ so that the trajectory passes a distance $D$ from the waypoint. For example, this mode could be used to ensure that the footprint of a sensor attached to a UAV passes over the waypoint. If $D \in [0, R \left(\frac{1}{\sin \beta} - 1\right)]$, then using Equation (19), it is straightforward to show that the proper choice of $\kappa$ 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 we wish to transition between waypoints with only one turn, then 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 $\kappa$ such that the path length of the $\kappa$-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 $\kappa$-trajectory.

**Lemma 6:** If $\kappa \in [0, 1]$, and $R = \hat{v}/c$, then the path length of the $\kappa$-trajectory shown in Figure 4 is given by

$$L(\kappa, w_{i+1}, w_i, w_{i-1}) = \|w_{i+1} - w_i\| + \|w_i - w_{i-1}\|$$

$$+ 2R \left(\frac{\pi - \beta}{2} + 2 \cos^{-1} (\Xi(\kappa, \beta)) - \Xi(\kappa, \beta) \sqrt{\frac{1}{\Xi(\kappa, \beta)^2} - 1 - (1 - \kappa) \cos \frac{\beta}{2} - \kappa \cot \frac{\beta}{2}}\right),$$

where

$$\beta = \cos^{-1} \left( \left( \frac{w_{i+1} - w_i}{\|w_{i+1} - w_i\|} \right)^T \left( \frac{w_i - w_{i-1}}{\|w_i - w_{i-1}\|} \right) \right),$$

and

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

**Proof:** With reference to Figure 9 we see that

$$L = \|w_{i+1} - w_i\| + \|w_i - w_{i-1}\| - 2(\overline{w_i c + \overline{e}}) + 2 \left( R \theta + R(\pi/2 + \theta - \beta/2) \right)$$

$$= \|w_{i+1} - w_i\| + \|w_i - w_{i-1}\| + 2R \left( \frac{\pi - \beta}{2} + 2\theta - \frac{\overline{w_i c + \overline{e}}}{R} \right).$$

We note that

$$\overline{e} = \sqrt{(R + \overline{cd})^2 - R^2}.$$  

By the law of sines,

$$\frac{\overline{w_i c}}{\sin \left(\frac{\pi}{2} + \theta - \frac{\beta}{2}\right)} = \frac{R + \overline{w_i p(\kappa)}}{\sin \left(\frac{\pi}{2} - \theta\right)} = \frac{R - \overline{cd}}{\sin \left(\frac{\pi}{2}\right)}.$$  

February 7, 2003
Solving for $\overline{cd}$ and $\overline{w_i c}$, we have

$$\overline{cd} = R - (R + \overline{w_i p(\kappa)}) \frac{\sin \left( \frac{\beta}{2} \right)}{\cos \theta},$$

(36)

$$\overline{w_i c} = (R + \overline{w_i p(\kappa)}) \frac{\cos \left( \theta - \frac{\beta}{2} \right)}{\cos \theta} = (R + \overline{w_i p(\kappa)}) \left( \cos \frac{\beta}{2} + \tan \theta \sin \frac{\beta}{2} \right).$$

(37)

From Figure 9, using $\cos \theta = \frac{R}{R + \overline{cd}}$, we get

$$\cos \theta = \frac{R + (R + \overline{w_i p(\kappa)}) \sin \left( \frac{\beta}{2} \right)}{2R}.$$  

(38)

Noting that $\beta \in (0, \pi)$ and substituting from Equation (19) gives

$$\overline{w_i p(\kappa)} = \kappa R \left( \frac{1}{\sin \left( \frac{\beta}{2} \right)} - 1 \right).$$
Solving for $\theta$ gives

$$\theta = \cos^{-1}\left(\frac{1 + \kappa + (1 - \kappa) \sin \frac{\beta}{2}}{2}\right) = \cos^{-1}\left(\Xi(\kappa, \beta)\right).$$  \hspace{1cm} (39)

Substituting Equations (36) and (38) into Equation (34), results in

$$\overline{ce} = R \left[\sqrt{\frac{4}{\left(1 + \kappa + (1 - \kappa) \sin \frac{\beta}{2}\right)^2}} - 1\right] = R \sqrt{\frac{1}{\Xi^2(\kappa, \beta)}} - 1.$$  \hspace{1cm} (40)

Noting from Figure 9 that $\tan \theta = \frac{\overline{ce}}{R}$ and using (40), we can solve for $w_{i+1}$ from Equation (37) as

$$w_{i+1} = R \left(\Xi(\kappa, \beta) - 1\right) \sqrt{\frac{1}{\Xi^2(\kappa, \beta)}} - 1 + (1 - \kappa) \cos \frac{\beta}{2} + \kappa \cot \frac{\beta}{2}. \hspace{1cm} (41)$$

Therefore substituting Equations (39), (40), and (41) into Equation (33) gives Equation (30).

From Lemma 6, it is clear the path length difference between the waypoint path $w_{i-1}, w_i, w_{i+1}$ and the associated $\kappa$-trajectory is given by

$$\Lambda(\kappa, w_{i+1}, w_i, w_{i-1}) = 2R \left(\frac{\pi - \beta}{2} + 2 \cos^{-1}\left(\Xi(\kappa, \beta)\right) - \Xi(\kappa, \beta) \sqrt{\frac{1}{\Xi^2(\kappa, \beta)}} - 1 - (1 - \kappa) \cos \frac{\beta}{2} - \kappa \cot \frac{\beta}{2}\right). \hspace{1cm} (42)$$

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

**Lemma 7:** If $\beta \in [0, \pi)$ and $\kappa \in [0, 1]$, then $\Lambda$ is a decreasing function of $\kappa$. In addition, $\Lambda(0, w_{i+1}, w_i, w_{i-1}) > 0$ and $\Lambda(1, w_{i+1}, w_i, w_{i-1}) < 0$.

**Proof:** Differentiating Equation (42) with respect to $\kappa$ gives, after some algebra,

$$\frac{\partial \Lambda}{\partial \kappa} = 2R \left[ - \frac{\partial \Xi}{\partial \kappa} \sqrt{\frac{1}{\Xi^2}} - 1 + \frac{\partial \Xi}{\partial \kappa} \sqrt{\frac{1}{1 - \Xi^2}} \left(-2 + \frac{1}{\Xi}\right) + \cot \frac{\beta}{2} \left(\sin \frac{\beta}{2} - 1\right)\right]. \hspace{1cm} (43)$$

From Equation (32) it is clear that $\beta \in [0, \pi)$ and $\kappa \in [0, 1]$ implies that $\Xi \geq 0$ and $\frac{\partial \Xi}{\partial \kappa} = (1 - \sin \frac{\beta}{2})/2 > 0$. Therefore if $-2 + 1/\Xi \leq 0$, then $\frac{\partial \Lambda}{\partial \kappa} < 0$. By definition we have

$$-2 + \frac{1}{\Xi} = -2 + \frac{1 + \sin \frac{\beta}{2}}{2} + \kappa \frac{1 - \sin \frac{\beta}{2}}{2} \leq -2 + \frac{1 + \sin \frac{\beta}{2}}{2} + \frac{1 - \sin \frac{\beta}{2}}{2} = -1.$$

Therefore $\Lambda$ is a decreasing function of $\kappa$.\[\square\]
Noting that $\Xi(0, \beta) = (1 + \sin \frac{\beta}{2})/2$, from Equation (42) we have after some algebra

$$\Lambda(0, \beta) = 2R \left[ \frac{\pi - \beta}{2} + 2 \cos^{-1} \left( \frac{1 + \sin \frac{\beta}{2}}{2} \right) - \sqrt{1 - \frac{1}{4} (1 + \sin \frac{\beta}{2})^2 - \cos \frac{\beta}{2}} \right].$$

Noting that $\Lambda(0, \pi) = 0$ and that

$$\frac{\partial \Lambda}{\partial \beta/2}(0, \beta) = 2R \left[ \left( \sin \frac{\beta}{2} - 1 \right) + \frac{\cos \frac{\beta}{2} \left( \frac{1}{4} (1 + \sin \frac{\beta}{2}) - 1 \right)}{\sqrt{1 - \frac{1}{4} (1 + \sin \frac{\beta}{2})^2}} \right],$$

we have by the Lagrange Remainder Theorem [29] that there exists an $a \in (\beta, \pi)$, such that

$$\Lambda(0, \beta) = \frac{\partial \Lambda}{\partial \beta/2}(0, a) \left( \beta - \pi \right).$$

It is straightforward to show that $\frac{\partial \Lambda}{\partial \beta/2}(0, a) < 0$ for all $a \in [0, \pi)$, which implies that $\Lambda(0) > 0$.

Noting that $\Xi(1, \beta) = 1$, we get from Equation (42) that

$$\Lambda(1, \beta) = 2R \left[ \frac{\pi - \beta}{2} - \cot \frac{\beta}{2} \right].$$

Differentiating we get

$$\frac{\partial \Lambda}{\partial \beta/2}(1, \beta) = \cot^2 \frac{\beta}{2} > 0.$$

Therefore, by the Lagrange Remainder Theorem, there exists an $a \in (\beta, \pi)$ such that

$$\Lambda(1, \beta) = \cot^2 \frac{\beta}{2} \left( \frac{\beta - \pi}{2} \right),$$

which implies that $\Lambda(1, \beta) < 0$ for all $\beta \in [0, \pi)$.

For timing critical missions it is necessary that the path length of the trajectory generated by the DTS is identical to the path length of the original waypoint trajectory $P$. The next theorem shows that there exists a $\kappa \in [0, 1]$ such that the path lengths are equal.

**Theorem 8:** If $\Lambda(\kappa, w_{i-1}, w_i, w_{i+1})$ is given by Equation (42), where $\beta \in [0, \pi)$ is given by Equation (31) and $R = \dot{v}/c$, then there exists a unique $\kappa^* \in [0, 1]$ such that $\Lambda(\kappa^*) = 0$. Furthermore, the $\kappa$-trajectory corresponding to $\kappa^*$ has the same path length as the waypoint path $w_{i-1}w_iw_{i+1}$, i.e.,

$$L = \|w_i - w_{i-1}\| + \|w_{i+1} - w_i\|.$$

Furthermore, $\kappa^*$ can be efficiently found numerically using a bisection search algorithm to within $O(2^{-n})$ where $n$ is the number of function evaluations of $\Lambda$.

**Proof:** From Lemma 6 we know that $\Lambda$ is the difference in path length between the $\kappa$-trajectory and the waypoint path. Therefore if $\Lambda = 0$, then the path lengths are equal. From Lemma 7 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 $\kappa^*$ to within $2^{-1}$. Subsequent function calls further refine the estimate.
V. Implementation Issues

In this section we discuss several practical implementation issues that must be addressed to ensure robust behavior of the DTS algorithm when it is implemented on digital hardware. Real-time implementation requires that the differential equation (7)–(9) must 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 Figure 6. For example, suppose that the DTS is tracking the straight-line segment $\mathbf{w}_{i-1}\mathbf{w}_i$ and the algorithm is looking for the intersection of $\mathcal{C}_L$ with $\mathcal{C}_{p(\kappa)}$ in order to detect the switching time $t_2$, then we may have the scenario depicted in Figure 10. The circle $\mathcal{C}_L$ may not intersect the circle $\mathcal{C}_{p(\kappa)}$ exactly at the sample times. Therefore, we need a robust method for detecting circle and line intersections, that also indicates when the intersection has been missed.

![Diagram](image)

Fig. 10. Effect of fixed sample-rate on switching-time detection.

Toward that end, define the function

$$
\Psi(a, b) \triangleq \text{sign} \left\{ \left[ \begin{array}{c} a \\ 0 \end{array} \right] \times \left[ \begin{array}{c} b \\ 0 \end{array} \right] \cdot \hat{\mathbf{k}} \right\}
$$

$$
= \text{sign} [a_x b_y - b_x a_y]
$$

(44)

where $\hat{\mathbf{k}}$ is the unit vector pointing into the plane. The function $\Psi$ can be used for several purposes. First, if $\mathbf{q}_i$ and $\mathbf{q}_{i+1}$ are unit vectors along the vectors $\mathbf{w}_{i-1}\mathbf{w}_i$ and $\mathbf{w}_i\mathbf{w}_{i+1}$ as shown in Figure 4, then the direction of the turn is given by $\Psi(q_i, q_{i+1})$, where $\Psi(q_i, q_{i+1}) > 0$ indicates that the turn from the current path segment to the next path segment will be clockwise (a right-handed turn), and $\Psi(q_i, q_{i+1}) < 0$ indicates that the turn will be counterclockwise (a left-handed turn). The function $\Psi$ can also be used partition the $\mathbb{R}^2$ plane into two distinct halves as shown in Figure 11. Referring to Figure 6, we will discuss the robust detection of switching times $t_2$ through $t_5$. 

February 7, 2003

DRAFT
Switching time $t_2$. Recalling that the center of circle $C_L$ is $c_L$ and that the center of circle $C_{p(\kappa)}$ as $p(\kappa)$, it can be seen from Figure 12 that the distance between the center of $C_L$ and $C_{p(\kappa)}$ is given by $\|c_L - p(\kappa)\|$. 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 $\|c_L - p(\kappa)\|$ assuming that at some sample time this quantity is less than $2R$. However, if $\kappa \approx 1$, then it is possible that $\|c_L - p(\kappa)\| > 2R$ at the sample times both prior and after the intersection at time $t_2$ as shown in Figure 12. Therefore, it is also necessary to detect when $c_L$ has moved from the half plane $S_1$ to the half plane $S_2$. Guided by Figure 11, this can be accomplished by monitoring the sign of $\Psi (c_L - p(\kappa), R(\pi/2)q_i)$, where

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

If $\Psi (c_L - p(\kappa), R(\pi/2)q_i) > 0$ then $c_L \in S_1$, otherwise $c_L \in S_2$.

Switching time $t_3$. The switching time $t_3$ can also be detected via a half plane argument. As shown in Figure 13, $t_3$ occurs when the center of circle $C_R (c_R)$ transitions from $S_1$ to $S_2$, 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 location of $\mathbf{c}_L$ at switching time $t_3$ through the line that bisects the angle $\beta$. The unit vector that defines the bisector of $\beta$ is given by Equation (18). The reflection of a vector through the line described by $\bar{\mathbf{q}}$ is given by the Householder transformation [30]

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

Therefore, as shown in Figure 14, we have

\[
\mathbf{c}_L(t_4) = \mathbf{w}_i + H(\bar{\mathbf{q}})(\mathbf{c}_L(t_3) - \mathbf{w}_i).
\]

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 Figure 14, 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 $\mathbf{w}_i \mathbf{w}_{i+1}$. With reference to Figure 14, 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 $t_3$.

The final implementation issue is due to the fact that at switching time $t_5$, the DTS may not be aligned perfectly with the waypoint segment $\overline{w_iw_{i+1}}$, therefore setting $u = 0$ will cause the DTS to drift from the waypoint segment. During the straight-line section we need to have a tracking algorithm that cause the DTS to asymptotically track the waypoint segment, while still satisfying the constraint $-c \leq u \leq c$.

Referring to Figure 15, let $\psi^d_i = \tan^{-1}(q_{iy}/q_{iz})$ be the angle created by the waypoint path. To simplify
the development, we will shift the origin to \( w_{i-1} \) and rotate the data by \( \psi_i^d \). Accordingly we have
\[
\begin{align*}
    e &= \begin{pmatrix} e_\parallel \\ e_\perp \end{pmatrix} = R(\psi_i^d)(z - w_{i-1}) \\
    \dot{e} &= \begin{pmatrix} \dot{e}_\parallel \\ \dot{e}_\perp \end{pmatrix} = \begin{pmatrix} v \cos(\psi - \psi_i^d) \\ v \sin(\psi - \psi_i^d) \end{pmatrix}.
\end{align*}
\]

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 9:** Given the nonlinear system
\[
\begin{align*}
    \dot{e}_\perp &= v \sin(\psi - \psi_i^d) \\
    \dot{\psi} &= u
\end{align*}
\]
with control constraint \( u \in [-c, c] \) where \( v \) is a positive constant. If
\[
u = -\text{sat}_c \left[ \frac{\cos^3(\psi - \psi_i^d)}{\gamma} \left( \frac{k_1 e_\perp}{\sqrt{e_\perp^2 + v^2}} + k_2 \sin(\psi - \psi_i^d) \right) \right],
\]
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:** Since for all \( \psi \in \Omega \), \( \cos(\psi - \psi_i^d) > 0 \), the change of variables
\[
\begin{align*}
    \zeta_1 &= e_\perp / v \\
    \zeta_2 &= \sin(\psi - \psi_i^d)
\end{align*}
\]
is a diffeomorphism resulting in the transformed system
\[
\begin{align*}
    \dot{\zeta}_1 &= \zeta_2 \\
    \dot{\zeta}_2 &= \sqrt{1 - \zeta_2^2} u,
\end{align*}
\]
and the transformed control law
\[
u = -\text{sat}_c \left[ \frac{(1 - \zeta_2^2)^{3/2}}{\gamma} \left( \frac{k_1 \zeta_1}{\sqrt{\zeta_1^2 + 1}} + k_2 \zeta_2 \right) \right].
\]
In the transformed coordinates \( \Omega = (-\infty, \infty) \times (-1, 1) \). Consider the value function
\[
V(\zeta) = k_1 \sqrt{\zeta_1^2 + 1} + 1 + \gamma \left( \frac{\zeta_2^2}{2(1 - \zeta_2^2)} \right)
\]
and note that \( V \) is positive definite on \( \Omega \) and goes to infinity as \( \zeta \) approaches the boundary of \( \Omega \). Differentiating (47) gives
\[
\dot{V} = \zeta_2 \left( \frac{k_1 \zeta_1}{\sqrt{\zeta_1^2 + 1}} + \frac{\gamma u}{(1 - \zeta_2^2)^{3/2}} \right).
\]
If \( u \) is not saturated, then direct substitution gives
\[
\dot{V} = -k_2 \zeta_2^2.
\]

Consider the case when \( u \) is positively saturated, then
\[
\dot{V} = \zeta_2 \left( \frac{k_1 \xi_1}{\sqrt{\xi_1^2 + 1}} + \frac{\gamma c}{(1 - \zeta_2^2)^{3/2}} \right),
\]
and \( u = c \) implies that
\[
-\frac{(1 - \zeta_2^2)^{3/2}}{\gamma} \left( \frac{k_1 \xi_1}{\sqrt{\xi_1^2 + 1}} + k_2 \zeta_2 \right) > c.
\]
Rearranging we have
\[
\frac{k_1 \xi_1}{\sqrt{\xi_1^2 + 1}} + \frac{\gamma c}{(1 - \zeta_2^2)^{3/2}} < 0.
\]
This inequality is trivially satisfied if \( \xi_1 \geq 0 \). Alternatively, if \( \xi_1 < 0 \) then
\[
-\frac{k_1 \xi_1}{\sqrt{\xi_1^2 + 1}} < k_1 < \gamma c < \frac{\gamma c}{(1 - \zeta_2^2)^{3/2}}.
\]
Since a similar argument holds when \( u \) is negatively saturated, we have shown that \( \dot{V} \) is negative semidefinite and that
\[
\hat{\Omega} = \{ \zeta \in \Omega : \dot{V} = 0 \} = \{ \zeta \in \Omega : \zeta_2 = 0 \}.
\]
On \( \hat{\Omega} \) the dynamics reduce to
\[
\dot{\zeta}_1 = 0
\]
\[
\dot{\zeta}_2 = -\text{sat}_c \left[ \frac{k_1 \xi_1}{\gamma \sqrt{\xi_1^2 + 1}} \right],
\]
which forces the system trajectories to leave \( \hat{\Omega} \) unless \( \zeta_1 = 0 \). Therefore, the largest invariant set in \( \hat{\Omega} \) is the origin, and the theorem follows by LaSalle’s invariance principle [31], [32].

The phase portrait of system (45) with the control (46) is shown in Figure 16 for several values of \( \gamma \), \( k_1 \) and \( k_2 \).

VI. Simulation

The \( \kappa \)-path 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 simulated vehicle was in a turn, the average run time for one time step was approximately 39 \( \mu s \), while if the vehicle was on a straight segment of the path, the average run time for one step was
approximately 16 $\mu$s. The computational simplicity of the algorithm enables its implementation in UAV applications where computational resources are modest.

Simulation results for the $\kappa$-path algorithm are shown in Figures 17 for minimum-time transitions ($\kappa = 1$), transitions through the waypoint ($\kappa = 0$), and transitions matching the length of the original straight-line path ($\kappa$ 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 Figure 17, it can be seen that the location of the center of the turn circle corresponding to $\kappa = 1$ varies depending on the $\beta$ angle 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 $\kappa$ values for each turn that result in equal path lengths depend on the $\beta$ angle between adjoining segments. For the last four turns of the equal-length path the $\kappa$ values were 0.266, 0.136, 0.406, 0.374. Typically, the more acute angles have smaller $\kappa$ values to equalize the length. Figure 18 shows close-up views of paths for waypoint segments joining at an acute angle. The differences in the minimum-time, through-the-waypoint, and equal-length transitions can be clearly seen.
VII. Conclusions

A method for generating point-constrained, time-extremal trajectories (called $\kappa$-paths) for transitioning between successive path segments of a waypoint path 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 deriving a $\kappa$-path 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 flies 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.

Acknowledgments

This work was partially funded by AFOSR grants F49620-01-1-0091 and F49620-02-C-0094, and by DARPA grant NBCH1020013.

References

Fig. 18. Sample trajectories – close up.


