Article

The International Journal of Robotics Research
2015, Vol. 34(3) 357–377

DOI: 10.1177/0278364914558017
ijr.sagepub.com
SAGE

Motion primitives and 3D path planning for fast flight through a forest

Aditya A. Paranjape[1], Kevin C. Meier[2], Xichen Shi[3], Soon-Jo Chung[3] and Seth Hutchinson[2]

Abstract

This paper presents two families of motion primitives for enabling fast, agile flight through a dense obstacle field. The first family of primitives consists of a time-delay dependent 3D circular path between two points in space and the control inputs required to fly the path. In particular, the control inputs are calculated using algebraic equations which depend on the flight parameters and the location of the waypoint. Moreover, the transition between successive maneuver states, where each state is defined by a unique combination of constant control inputs, is modeled rigorously as an instantaneous switch between the two maneuver states following a time delay which is directly related to the agility of the robotic aircraft. The second family consists of aggressive turn-around (ATA) maneuvers which the robot uses to retreat from impenetrable pockets of obstacles. The ATA maneuver consists of an orchestrated sequence of three sets of constant control inputs. The duration of the first segment is used to optimize the ATA for the spatial constraints imposed by the turning volume. The motion primitives are validated experimentally and implemented in a simulated receding horizon control (RHC)-based motion planner. The paper concludes with inverse-design pointers derived from the primitives.

Keywords

Aerial robotics, online path planning, flight control, motion primitives, optimal control, bio-inspired flight

1. Introduction

Birds flying through dense forests represent a combination of agile airframes and adroit motion planners capable of ensuring collision-free flight at high speeds in obstacle-rich environments. The motivation for this paper is the prospect of replicating the capability of birds to ensure that unmanned fixed wing aircraft can fly rapidly through a dense obstacle field such as a forest. Our recent paper (Paranjape et al., 2013a) showed how to control the turning flight using wing articulation which is present naturally in flapping wings.

There are multiple challenges to flying a robotic aircraft at high speeds in a densely crowded field: localization and navigation; online path planning; and determining the control inputs required to follow a path demanded by the path planner. The control design challenge is particularly exacerbated in fixed-wing aircraft, whose dynamics are highly nonlinear and the aerodynamics are rife with significant structural and parametric uncertainties, particularly in flight regimes which are desirable for rapid maneuvering. From a purely motion planning (as against control design) perspective, the flight speed envelope of fixed wing robotic aircraft is restricted and, importantly, bounded from below by a significant positive value (i.e. the stall speed of the airframe).

This paper focuses primarily on the control design problems. We assume that: (1) an existing path planner, which is aware of the limitations of the airframe and the workings of the control system, chooses waypoints along the path; and (2) the robotic aircraft is equipped with a vision-based navigation system or lidar which provides, among other things, the bearing and distance to a waypoint. Such vision-based navigation and localization systems are well-established in

[1]Department of Mechanical Engineering, McGill University, Canada
[2]Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, USA
[3]Department of Aerospace Engineering, University of Illinois at Urbana-Champaign, USA

Corresponding author:
Soon-Jo Chung, Department of Aerospace Engineering, University of Illinois at Urbana-Champaign, 104 S. Wright, MC236, 306 Talbot Laboratory Urbana, IL 61801, USA.
Email: sjchung@illinois.edu

the literature (Langelaan and Rock, 2005; Celik et al., 2009; Bry et al., 2012; Dani et al., 2013; Yang et al., 2013). In this paper, we solve two problems.

1. The first problem is a standard two-point boundary value problem: given a waypoint, determine the control inputs required to fly the robotic aircraft to that waypoint. A key challenge here is to make the control determination formula or algorithm as simple as possible for computation efficiency, while accurately accommodating the nonlinear dynamics of the robotic aircraft and the dynamics of the control actuators.
2. The second problem is particularly relevant for high-speed flight through an obstacle-rich field: if the path planner is unable to identify a suitable waypoint, determine the control inputs required to reverse the heading of the robotic aircraft inside the available volume.

We design two families of motion primitives, one for each problem, which are continuously parametrized in the space of control inputs. The first family of primitives consists of *steady* 3D turns designed for normal forward flight between two prescribed points in space. The second family of primitives consists of *instantaneous* 3D turns accomplished using a sequence of constant control inputs, which are referred to as aggressive turn-around (ATA) maneuvers. Unlike the steady turns, which guide the robotic aircraft between two waypoints, the ATA maneuver allows a robotic aircraft flying at high speeds to reverse its heading inside a small volume of space. It is meant to allow the robot to back-track safely if it approaches an impenetrable pocket of obstacles. The net effect of including the ATA family is that it allows the aircraft to operate safely at much higher speeds than it could otherwise (see, e.g., Karaman and Frazzoli, 2012).

The motion primitives designed in the paper enable efficient online path planning by providing algebraic solutions to the two-point boundary value problem of determining the control inputs required to steer a robotic aircraft between two points in space. The motion primitives explicitly accommodate the dynamics of the robotic aircraft, using the internal time-scale separation and time-delay-based model of agility, while providing a practical way to enable fixed-wing robotic aircraft to back-track along their path without performing a stop-and-U-turn maneuver which was hitherto possible only in quadrotors and helicopters.

1.1. Literature review

The problem of flying a robotic aircraft through an obstacle field falls within the ambit of robotic motion planning in the presence of differential constraints. Well-known methodologies for solving this problem include those based on state-space sampling, mathematical programming, potential function-based solutions, and decoupled trajectory and path planning (see Goerzen et al. (2010) for an extensive review of these methods from the perspective of unmanned aerial vehicles).

The motion primitives-based approach to motion planning is particularly appealing because the path planner relies on very limited knowledge about the aerial vehicle beyond the kinematic constraints expressed in the form of motion primitives. Therefore, for example, a single path planner can be used for a large class of off-the-shelf robotic aircraft equipped with their own autopilots and control systems. Conversely, the path planning algorithm can be chosen from a wide range of methods such as probabilistic road maps (PRM) (Kavraki et al., 1996), rapidly exploring random trees (RRT) (Frazzoli et al., 2002; Schouwenaars et al., 2003; Frazzoli et al., 2005; Kehoe et al., 2006; LaValle, 2006), and model predictive control (Gray et al., 2012).

A motion primitive is defined via a known sequence of control inputs which results in well-characterized motion. For example, a turn in 3D space consists of constant elevator, aileron, rudder and thrust settings, with each control combination defining a unique combination of turn rate, flight speed, and flight path angle (rate of climb or descent). On the other hand, an aggressive maneuver may require a sequence of control inputs, even through active state feedback, e.g. aggressive heading reversals proposed by Matsumoto et al. (2010).

The motion primitives are provided to the path planner as a library, and a trajectory generated by the path planner is deemed to be feasible if it can be written as a concatenation of trajectories from the library (Frazzoli et al., 2002; Schouwenaars et al., 2003; Frazzoli et al., 2005) and if it satisfies prescribed collision-avoidance constraints, which may include safety margins to accommodate modeling and parametric uncertainties in the flight dynamics. In addition to motion primitives used for nominal flight (i.e. straight flight and turns), it is possible to use certain rapid transitory maneuvers, as illustrated by Schouwenaars et al. (2004) for the case where the underlying maneuver automaton failed to find a feasible cruising solution. A major drawback of using a library consisting of only finitely many control combinations as primitives is that it potentially rules out a large set of otherwise flyable paths. Another drawback is that the process of constructing a sequence of primitives for flying between successive waypoints while ensuring compatibility between the primitives can be computationally tedious when picking primitives from a library. In contrast, Dever et al. (2006) presented a general motion planning framework using continuously parametrized motion primitives, where numerical interpolation among a continuously parametrized family of motion primitives, together with boundary condition matching, yielded the desired trajectory and the set of control inputs required to fly the trajectory.

1.2. Contributions

This paper aims to design two families of motion primitives which are continuously parametrized in the space of control inputs (as against continuously parametrized with respect to the flight time with a finite number of control combinations (Frazzoli et al., 2005)) for agile flight through a dense obstacle field. The first family of primitives consists of steady 3D turns between two given points in the space, while the second family of primitives consists of transient ATA maneuvers for achieving an almost instantaneous reversal of heading. The notion of airframe agility is rigorously captured in the switching logic between successive primitives via a time-delay formulation. The contributions of the paper are as follows.

1. Analytical formulae, in the form of algebraic relations, are derived for control inputs required to accomplish a circular 3D turn between two points in space, subject to the performance limitations of the robotic aircraft. These formulae are motivated by pure pursuit laws for missiles and large aircraft (Ollero and Heredia, 1995; Park et al., 2007; Berg-Taylor et al., 2008), and were presented by the present authors in Paranjape et al. (2013b). A formal, analytical approach is presented to account for limited airframe agility, wherein the finite agility is modeled as a non-zero value of the time required to switch between successive control inputs. This forms the basis of the stitching logic for the motion primitives presented in the paper.
2. An ATA maneuver is designed to help the motion planner deal with localized impenetrable pockets of obstacles (see Figures 1 and 2). The ATA maneuver, first presented by the present authors in Paranjape et al. (2013c), is an *instantaneous* 3D turn with a sequence of constant control inputs, and with the time delay between the inputs acting as an additional design parameter. The ATA maneuver could help increase the speeds at which aircraft can fly safely through dense obstacle fields. The ATA maneuver primitive is derived offline and the only online computation required is the choice of the time delay, based on the sensed shape of the turning volume.
3. The aforementioned primitives are demonstrated experimentally through indoor flight tests. They are also incorporated into a receding horizon (or model predictive) control (RHC)-based motion planner whose capabilities are demonstrated by simulation. The motion planning algorithm used in this paper is similar to PRM in that at every point it chooses, from amongst a randomly generated sample of waypoints in the visible region, a feasible waypoint which minimizes a prescribed cost function. However, unlike PRM, and since the environment is unknown, the path planning is done locally as increasingly more information about the environment becomes available as the flight progresses.

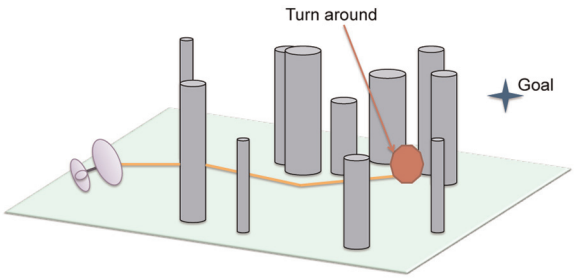

Fig. 1. Situation where an aggressive turn is mandated.

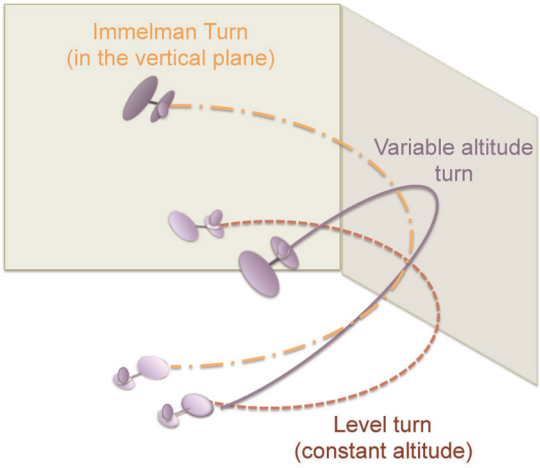

Fig. 2. Schematic of ATA maneuvers.

4. Simulation results are used to assess the maximum speed at which the aircraft can navigate through the forest, as a function of the tree density, without getting trapped in inescapable ATA loops.

The derivation of the closed-form formulae for the trajectory between successive waypoints, which is used to compute the constant control inputs required to fly it, brings with it additional benefits. The values of the control inputs can be used in the cost function for optimizing the path (see Section 5). Moreover, the analytical expression for the trajectory connecting successive waypoints can be used to compute the distance of the trajectory from nearby obstacles, and assess the feasibility of the trajectory quickly. Finally, since expressions for the control inputs as well as the trajectories are in the form of closed-form algebraic equations (in contrast to online optimization or numerical interpolation approaches used in the literature), their computation is simple and computationally light, which frees up computational resources for tasks such as sensing and mapping.

Karaman and Frazzoli (2012) computed an upper bound on the flight speed above which collision-free flight was almost surely impossible, as well as a lower bound below which an infinite number of collision-free trajectories were guaranteed to exist. The work presented in this paper, despite some commonality in spirit, is different in that our

notion of a safe flight speed is less restrictive; flight speeds are considered safe only when the possibility of the so-called ATA traps is negligible (aside from collision avoidance). ATA traps are situations where the only admissible maneuvers are a repeating sequence of ATAs, essentially blocking the aircraft inside a spatial pocket (see Section 5.2)

Some of the material presented in this paper was presented previously at conferences (Paranjape et al., 2013b,c). A summary of the major changes is in order.

1. In Section 5.1, we have added an analytical derivation of the maximum expected deviation of the aircraft trajectory about the primitive, which yields a value for the threshold distance used for assessing the admissibility of a candidate trajectory.
2. The derivation of the control laws is altogether new vis-á-vis the prior publications. It may be found, together with the calculation of the deviation about the primitive, in Section 6.1 and Appendix B.
3. We have added extensive experimental results in this paper. We have also included simulation results which show the dependence of flight time on the flight speed, and the maximum flyable speed on the density of the forests (Section 5.2).

The paper is organized as follows. The equations of motion are derived in Section 2. Control laws for steady 3D turns (also called *routine* flight) are derived in Section 3, together with an analytical approach for accommodating the agility of the robotic aircraft. Aggressive turns are modeled in Section 4, and the motion planning algorithm is described in Section 5 together with simulation results. Experimental results are presented in Section 6, while in Section 7, the analysis of the aforementioned sections is used to derive design pointers for robotic aircraft intended for missions involving high-speed flight in forest-like environments.

2. Equations of motion and inner-loop control

We will state the equations of motion in the standard form found in the literature on flight mechanics (Kelley and Edelbaum, 1970; Kelley, 1971). In Table 1, we have introduced the commonly used symbols in flight mechanics. The angle of attack α is defined as the angle made by the longitudinal axis of the aircraft (the axis that passes through the rear tip of the fuselage and the nose of the aircraft) with the projection of the velocity vector onto the plane of symmetry of the aircraft. The wind axis roll angle μ is the complement of the angle made by the lift vector with the *global* horizontal plane. The wind axis roll angle μ differs from the body axis bank angle (denoted by ϕ and also referred to as the body roll angle), in that it is given by $\sin\mu = \sin\phi\cos(\gamma + \alpha)$, where γ is the flight path angle which measures the inclination of the velocity vector with respect to the global horizontal plane. The angles α and μ have been depicted in Figure 3.

Table 1. Nomenclature.

Symbol	Explanation
$C_L(\alpha), C_D(\alpha)$	coefficients of lift and drag
T	thrust per unit mass
V	flight speed
α	angle of attack
γ, χ	flight path angle, heading angle
μ	wind axis roll angle

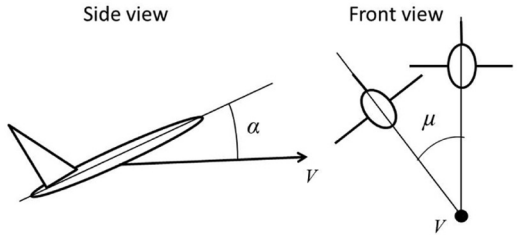

Fig. 3. Angle of attack α and wind axis roll angle μ depicted schematically, with V denoting the velocity vector (coming out of the plane of the paper in the front view).

The complete equations of motion of a rigid aircraft are given in Appendix B. The complete state of a rigid aircraft is described by two sets of variables. The first set of variables, called the *outer states*, consists of the position coordinates x, y, z and the velocity vector of the robotic aircraft (V, γ, χ). Note that velocity vector has been described in terms of its magnitude and two angles which define its orientation with respect to a ground-fixed inertial frame of reference. The second set of variables, called the *inner states*, consists of the Euler angles and angular velocity vector of the robotic aircraft. These two sets are not decoupled. Rather, there is a time-scale separation between them: the dynamics of the inner states are an order of magnitude faster than those of the outer states. We will ignore the dynamics of the inner states while deriving the motion primitives with the understanding that they can be controlled adequately by inner-loop controllers (see Section 6.1 and Figure 4).

For the dynamics of the outer states, the thrust T_c, angle of attack α_c, and the wind axis roll angle μ_c act as the control inputs. The motion primitives are defined in terms of the outer states whose dynamics are described presently. To simplify the notation, define

$$k = \frac{\rho S}{2m}, \quad T = \frac{\text{Thrust}}{m} \tag{1}$$

where ρ is the density of air, S is the area of the wing (a reference area), and m denotes the mass of the aircraft. Note that k is the scaled inverse of the wing loading mg/S, where g is the gravitational constant. The outer-state dynamics are

then described by the following equations (Kelley and Edelbaum, 1970; Kelley, 1971):

$$\begin{aligned}
&\dot{x} = V\cos\gamma\cos\chi, \quad \dot{y} = V\cos\gamma\sin\chi, \quad \dot{h} = V\sin\gamma \\
&\dot{V} = \left(T\cos\alpha - kV^2 C_D(\alpha)\right) - g\sin\gamma \\
&\dot{\gamma} = \left(\frac{T\sin\alpha}{V} + kVC_L(\alpha)\right)\cos\mu - \frac{g\cos\gamma}{V} \\
&\dot{\chi} = \left(\frac{T\sin\alpha}{V} + kVC_L(\alpha)\right)\frac{\sin\mu}{\cos\gamma}
\end{aligned} \tag{2}$$

where h denotes the altitude of the robot. The thrust T, angle of attack α and bank angle μ are related to the commanded values T_c, α_c, and μ_c through the first-order equations

$$\dot{T} = a_T(T_c - T), \quad \dot{\alpha} = a_\alpha(\alpha_c - \alpha), \quad \dot{\mu} = a_\mu(\mu_c - \mu) \tag{3}$$

where $a_{\{\cdot\}}$ denote the inverses of the time constants. The behavior in (3) is achieved with the help of inner-loop controllers for α and μ. The motion planning problem involves choosing waypoints and mapping their choice to T_c, α_c, and μ_c.

The angle of attack is controlled directly by deflecting the elevator, a flap located on the horizontal tail of the aircraft. If the α-dynamics of an aircraft (given in Appendix B) are stable and show desirable convergence properties, it may suffice to use a feed-forward signal for the elevator deflection: $\delta_e = f(\alpha_c)$, where the function $f(\cdot)$ can be determined either from high-fidelity models or from flight tests, as described in Section 6.1.

The control of the wind axis roll angle μ is a coupled roll–yaw control problem which involves regulating the sideways motion of the aircraft (in particular, the angle of sideslip, β, whose dynamics is given in Appendix B, is usually regulated at zero) while controlling the bank angle of the aircraft. Roll–yaw control is provided by the ailerons and the rudder, which are located on the main wing and the vertical tail, respectively. In some aircraft, such as the one used for the experiments described in this paper, the ailerons may be absent. In such cases, the rudder is used for ensuring that the aircraft rolls through the appropriate angle (μ), but the sideways motion itself is not explicitly regulated. A similar roll–yaw control problem was solved in Paranjape et al. (2013a), where wing articulation provided a primarily yaw-based control action. The design of inner-loop controllers which actuate the rudder and the elevator for controlling μ and α has been addressed for an experimental aircraft in Section 6.1 and follows a similar approach as in Paranjape et al. (2013a). We note here that the control law for the rudder deflection δ_r is of the form

$$\begin{aligned}
&r_c = r_c(\mu_c, V), \quad \text{a known feed-forward mapping,} \\
&\delta_r = k_p(r_c - r) + k_I\int_0^t (r_c - r)\,dt
\end{aligned} \tag{4}$$

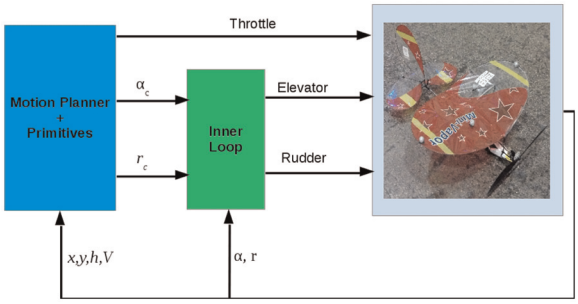

Fig. 4. Two-stage control system for the robotic aircraft. The motion primitives derived in Sections 3 and 4 constitute the "outer loop" controller.

where r is the body axis yaw rate, and r_c denotes the commanded yaw rate. The proportional and integral gains k_p, $k_I > 0$ are designed on a case-by-case basis. The derivation of the feed-forward mapping $r_c(\mu_c, V)$ has been explained in Section 6.1.

A final note concerns time delays in the sequel. We will encounter two different time delays: the first, denoted by τ_a, is the time spent until an instantaneous switch from one maneuver state to another, and captures the finite time required to perform a transition between the maneuver states (Section 3) in the presence of internal dynamics; the second, τ_d, will denote the time, after the commencement of the ATA, when the aircraft starts to roll into the turn (Section 4).

3. Mapping end points to control inputs: The agility connection

In this section, we derive an algebraic formula which maps the distance and the bearing of the desired waypoint to the control input required to reach it, such that the dynamics of the vehicle (2) are not ignored in the process. This is a unique feature of our algorithm.

The waypoints are chosen inside a 3D visual sensing cone which is defined by placing the aircraft at its vertex, and by aligning the axis of symmetry of the cone with the instantaneous velocity vector. The length of the cone is bounded by the sensing radius.

We first make the notion of aircraft agility precise. We interpret agility as the ability to change accelerations rapidly, and therefore, define agility tentatively as the rate of change of acceleration for translational motion and rate of change of angular velocities for rotational motion (Paranjape and Ananthkrishnan, 2006). For example, the turn rate (which is the rate of change of the velocity vector and hence an acceleration) is changed by rotating the lift vector about the longitudinal (body x-)axis. Thus, the time required to rotate the lift vector through a prescribed angle is an important agility metric.

In this section, we will start with the assumption of unlimited agility (instantaneous rotation of the lift vector,

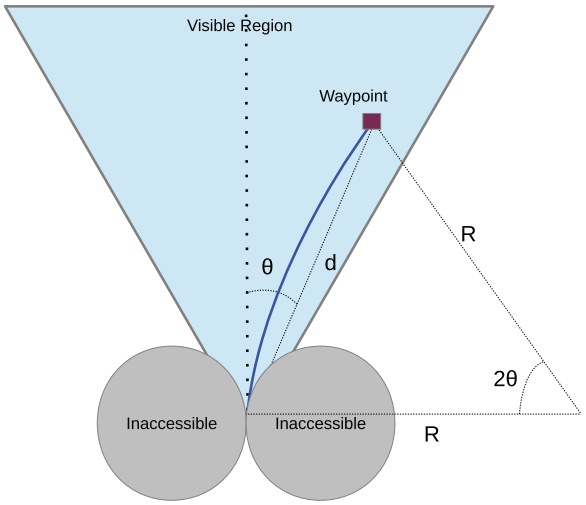

Fig. 5. Circular trajectory given an end point, and assuming infinite agility. It must be noted that the cone shown here is a 2D projection of a 3D visual sensing cone, and the circular trajectory is also the 2D projection of the 3D trajectory connecting the waypoints which may have different altitudes.

Section 3.1), and then use the results to analyze the case of finite agility (Section 3.2).

3.1. Unlimited agility

Consider the dynamics of χ from (2), given by

$$\dot{\chi} = \left(\frac{T \sin\alpha}{V} + kVC_L\right)\frac{\sin\mu}{\cos\gamma} \tag{5}$$

When the agility is infinite, it is possible to change $\dot{\chi}$ *instantaneously* between any two admissible values (including the limiting values), reflecting the ability to change μ and α instantaneously.

Suppose that the aircraft turns with a constant speed V. This assumption simplifies the derivation of the primitives considerably. While implementing the primitives in a practical setting, the value of V should be replaced by the velocity at the waypoint where the control input commands are calculated, if the flight speed is expected to remain more or less the same during consecutive segments. If the flight speed is expected to change significantly, the expected value of the average flight speed during the segment should be used for computing the control inputs (T_c, α_c, μ_c).

Consider Figure 5 which shows the x–y projection of the 3D sensing cone at an arbitrary instant of time. With a mild abuse of notation, we refer to the location of the aircraft at this instant of time as the current waypoint. The vertex of the cone coincides with the aircraft. Suppose that the robotic aircraft needs to reach the point (d, θ) shown in Figure 5, which is chosen as a candidate next waypoint by the motion planning algorithm. Note that the altitude of the next waypoint need not be the same as the current waypoint, but we will first address the motion in the projected x–y plane. The trajectory linking them can be parametrized by a single set of constant control inputs (T, α, μ). From Figure 5, we deduce that the turn radius is given by

$$R = \frac{d\cos\theta}{\sin 2\theta} = \frac{d}{2\sin\theta} \tag{6}$$

Since the turn radius is also given by $R = V\cos\gamma/\dot{\chi}$, it follows from (5) and (6) that the commanded value of μ for (3) satisfies

$$\sin\mu_c = \frac{2\sin\theta\cos^2\gamma}{\left(kC_L + \frac{T\sin\alpha}{V^2}\right)d} \tag{7}$$

We will now eliminate the term $kC_L + T\sin\alpha/V^2$. From the equation for $\dot{\gamma}$ in (2), it follows that we can choose the angle of attack α and thrust T to ensure that

$$kC_L + \frac{T\sin\alpha}{V^2} = \frac{1}{\cos\mu}\left(\frac{\dot{\gamma}_{\text{des}}}{V} + \frac{g}{V^2}\cos\gamma\right) \tag{8}$$

where $\dot{\gamma}_{\text{des}}$ denotes the desired value of $\dot{\gamma}$. We will derive an expression for $\dot{\gamma}_{\text{des}}$ later in this section. Substituting (8) into (7) gives the following expression for μ_c:

$$\tan\mu_c = \frac{2V^2\sin\theta\cos^2\gamma}{d(g\cos\gamma + V\dot{\gamma}_{\text{des}})} \tag{9}$$

If we assume that $\dot{\gamma}_{\text{des}} \ll (g/V)$, we get the following simplified expression:

$$\tan\mu_c = \frac{2V^2\sin\theta\cos\gamma}{gd} \tag{10}$$

If the value of μ_c is larger than the limiting value, it is possible to change the commanded flight path angle γ_c to compensate for the deficiency in μ. In general, we choose γ_c to ensure that aircraft reaches the waypoint at the desired altitude. We estimate the commanded flight path angle as $\gamma_c = \tan^{-1}((h_{\text{waypoint}} - h_{\text{current}})/d)$, where h_{current} is the altitude of the aircraft at the current waypoint. This is, in fact, the average value of the flight path angle over the complete segment. Let γ_{current} denote the flight path angle at the current waypoint. We choose $\dot{\gamma}_{\text{des}}$ as the average rate of change of $\dot{\gamma}$ over the complete segment: $\dot{\gamma}_{\text{des}} = 2(\gamma_c - \gamma_{\text{current}})/t_{\text{way}}$, where $t_{\text{way}} \approx d/V$ denotes the flight time between the current and the next waypoint. From (8), we choose α_c to satisfy

$$C_L(\alpha_c) = \frac{2V(\gamma_c - \gamma_{\text{current}})/t_{\text{way}} + g\cos\gamma_c}{kV^2\cos\mu_c} - \frac{T\sin\alpha_c}{kV^2\cos\mu_c} \tag{11}$$

The commanded value of thrust T_c is found by solving for $\dot{V} = 0$ in (2):

$$T_c = \frac{kV^2C_D(\alpha_c) + g\sin\gamma_c}{\cos\alpha_c} \tag{12}$$

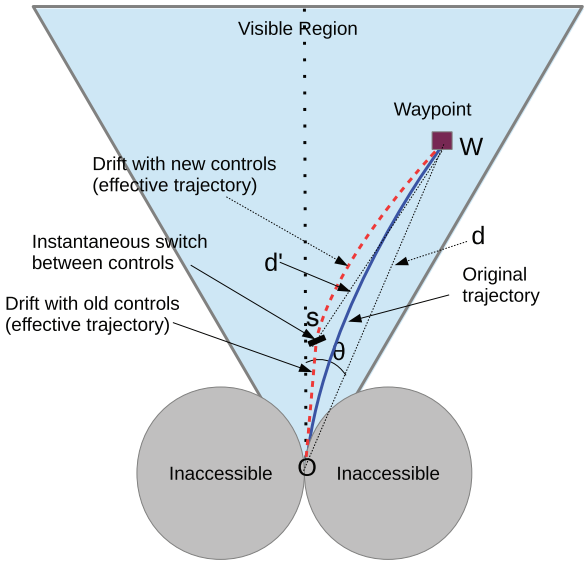

Fig. 6. Decomposing the original trajectory (solid blue) into a drift with the original control inputs and a circular trajectory with the new control inputs to the desired end point when the agility is finite (dashed red). The point O is the current waypoint, S denotes the switching point between the two sets of control inputs, and W denotes the waypoint.

Note that the thrust T is used in (11) with the assumption that the thrust will not change significantly during the time t_{way}.

The final note in this section concerns the case where the desired flight speed, V_{com}, in the segment between the two waypoints is considerably different from the speed at the first of the two waypoints, denoted V_1. In such cases, the thrust command T_c in (12) can be augmented by an additive term $k_T(V_{\text{com}} - V_1)$, where $k_T > 0$ can be chosen using the approach used for deriving the coefficient of $\gamma_c - \gamma$ in (11). We note, however, that the speed command should be held fixed to the extent possible because the phugoid $(V - \gamma)$ dynamics are usually slow and underdamped.

3.2. Finite agility using time-delay-based approach

Finiteness in agility is a consequence of the fact that α and μ both require a finite amount of time to change values. A well-designed inner-loop controller will ensure that the dynamics of α and μ behave like low-pass filters, as illustrated in (3).

A low-pass filter of the form $\frac{1}{\tau s+1}$ may be viewed as the first-order Padé approximation of a time delay τ (see Kuo and Golnaraghi, 2003, p. 183). Alternatively, one may formally map the response of a system coupled to a low-pass filter to that of the same system with a time-delayed input, and it can be shown that τ is indeed a suitable value of the time-delay that yields an identical steady-state response in both cases, at least for step inputs. This has been shown in Appendix C. It must be noted that we have actually solved an "inverse-Padé approximation" problem; i.e. given the rational transfer function model for the μ dynamics (Equation (3)), we have obtained the most suitable time-delay approximation for it.

Thus, we can model the agility of an aircraft via a time delay in the system. In particular, this allows us to decompose the trajectory of the aircraft, as it switches from one control input to another and flies from the vertex of the cone in Figure 6 to the waypoint located at a distance d and bearing θ from the vertex (labeled as the "original trajectory" in Figure 6), as the *sum* of two segments (labeled as the "effective trajectory"): (i) a drift with the initial control input μ_0 for time $\tau_a = 1/a_\mu$, where a_μ is a measure of the roll agility in Equation (3); and (ii) a drift along the new roll angle μ_c for the remainder of the time. The two segments take the aircraft to (d, θ) in the same time as the original trajectory, but do not coincide with the original trajectory. We seek to calculate μ_c given (d, θ).

We first note that the drift distance can be approximated by $V\tau_a$, and the aircraft may be assumed to turn through an angle $\dot{\chi}_0\tau_a$ during this time, where $\dot{\chi}_0$ is the initial turn rate. As long as τ_a is small, the initial drift distance may be approximated by that along a straight line segment connecting the initial point and the switching point between the two segments.

After the initial drift is complete, the controls switch to the new configuration; in particular, $\mu_0 \mapsto \mu_c$. We can now use the formulation from Section 3.1 after replacing (d, θ) with the new distance d' and bearing angle θ_n (see Figure 7).

From the quadrilateral $S'OCS$ in Figure 7, it is evident that $\angle SS'O = \pi - 2\nu$, so that $\angle S'OS = \angle S'SO = \nu$. Thus, it follows that $\theta_n = \pi - \nu_a - \nu$, where the angle ν_a is yet to be determined.

The new distance, d', and the angle ν are given by

$$d'^2 = d^2 + (V\tau_a)^2 - 2dV\tau_a\cos(\theta - \nu), \quad \nu = \frac{\dot{\chi}_0\tau_a}{2} \tag{13}$$

We calculate the angle ν_a using the cosine rule:

$$\cos\nu_a = \frac{(d')^2 + (V\tau_a)^2 - d^2}{2d'V\tau_a} = \frac{V\tau_a - d\cos(\theta - \nu)}{d'} \tag{14}$$

The new bearing is given by $\theta_n = \pi - \nu_a - \nu$, and we can use the formulation from the previous section with $(d, \theta) \leftarrow (d', \theta_n)$:

$$\tan\mu_c = \frac{2V^2\sin\theta_n\cos\gamma}{gd'} \tag{15}$$

The angle of attack and thrust commands (α_c and T_c) for (3) are chosen as described in (11) and (12).

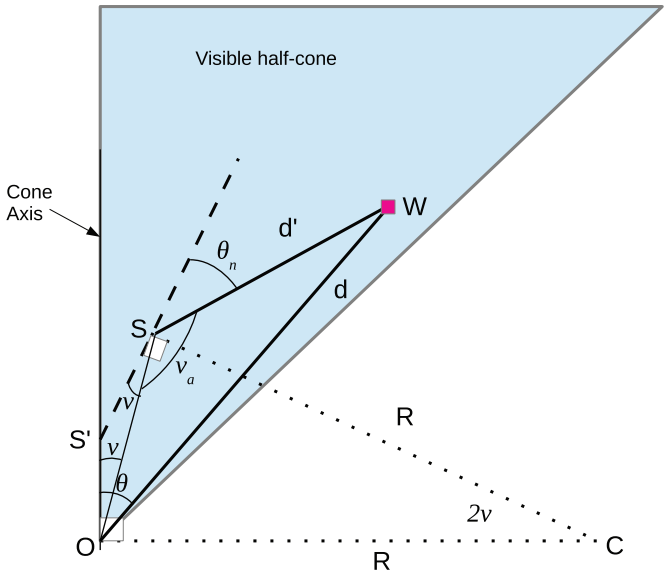

Fig. 7. Half cone showing a magnified view from Figure 6, as an aid to computing the distance and bearing to the waypoint W after the drift along the old control inputs. The point S denotes the switching point, while S' is the intersection of the velocity vector at S with the axis of the cone. The point C is the center of the circular arc which forms the drift trajectory with the old control inputs (see Figure 6).

4. Aggressive turn primitive

Aggressive turns are performed with the objective of reversing the aircraft heading, i.e. changing it by 180°, when collision-free forward flight is infeasible within the performance limitations of the aircraft (see Figure 1). The word "aggressive" also suggests that these maneuvers take the aircraft to the boundary of its flight envelope, and they are unsustainable (and, hence, purely transient) in nature. We assume that the sensing systems on board the aircraft can detect the obstacles around the turning volume in order to tune the ATA maneuver (in a sense which will become evident later in this section). The validity of this assumption can be ensured by designing the motion planning algorithm appropriately. However, a problem may arise in this regard if, for example, the ATA in question follows another ATA as a result of a delayed actuator response or adverse gusts which lead to a heading change in excess of the planned change of 180°. This limitation has not been addressed in the paper.

We design the ATA primitive systematically using an optimal control formulation. The optimal control problem is stated as follows:

$$\min_{T_c,\,\alpha_c,\,\mu_c} \eta_\gamma \gamma^2(t_f) + \int_0^{t_f} (\eta_x x^2 + \eta_y y^2 + \eta_h h^2 + \eta_T T_c^2)\, dt$$
$$\text{subject to the dynamics in (2) and (3) and}$$
$$\chi(t_f) - \chi(0) = \pi, \quad \mu(t_f) = 0$$
$$T_c \in [0,\, T_{\max}], \quad |\alpha_c| \le \alpha_{\max}, \quad |\mu_c| \le \mu_{\max} \tag{16}$$

where $0 < \alpha_{\max} \le \alpha_{\text{stall}}$. In this section, we assume that, $T_{\max} = 8$, $\alpha_{\max} = 35°$, and $\mu_{\max} = 60°$.

Note that the terminal time t_f is a free variable. The constraint $\mu(t_f) = 0$ and the penalty on the terminal flight path angle $\gamma(t_f)$ (in the form of $\eta_\gamma > 0$) ensure that the aircraft recovers to a wing-level flight condition with as straight a flight path as possible. The weights η_x, η_y and η_h are chosen to match the spatial constraints. The constraints on the control inputs are chosen to match real aircraft, such as the experimental testbed in Figure 15(a).

We will first attempt to solve the optimum control problem analytically to understand the structure of the ATA metric in terms of the control inputs required for it. It will transpire that the ATA maneuver can be viewed as a sequence of bang–bang control inputs. In order to identify the switching instants, we will solve the complete problem numerically.

Define the Hamiltonian

$$\begin{aligned} H = {} & \eta_x x^2 + \eta_y y^2 + \eta_h h^2 + \eta_T T_c^2 + \lambda_x V \cos\gamma \cos\chi \\ & + \lambda_y V \cos\gamma \sin\chi + \lambda_h V \sin\gamma \\ & + \lambda_V (T\cos\alpha - kV^2 C_D - g\sin\gamma) \\ & + \lambda_\gamma \left(\left(\frac{T\sin\alpha}{V} + kVC_L(\alpha) \right) \cos\mu - \frac{g\cos\gamma}{V} \right) \\ & \lambda_\chi \left(\frac{T\sin\alpha}{V} + kVC_L(\alpha) \right) \frac{\sin\mu}{\cos\gamma} + \lambda_\mu a_\mu (\mu_c - \mu) \\ & + \lambda_T a_T (T_c - T) + \lambda_\alpha a_\alpha (\alpha_c - \alpha) \end{aligned} \tag{17}$$

This gives us the following dynamical equations for the co-states

$$
\begin{aligned}
&\dot{\lambda}_x = -2\eta_x x, \quad \dot{\lambda}_y = -2\eta_y y, \quad \dot{\lambda}_h = -2\eta_h h \\
&\dot{\lambda}_V = -(\lambda_x \cos\gamma\cos\chi + \lambda_y \cos\gamma \sin\chi + \lambda_h \sin\gamma) \\
&\quad + 2\lambda_V kVC_D + \lambda_\gamma\left(\left(\frac{T\sin\alpha}{V^2 - kC_L}\right)\cos\mu - \frac{g\cos\gamma}{V^2}\right) \\
&\quad + \lambda_\chi\left(\left(\frac{T\sin\alpha}{V^2} - kC_L\right)\frac{\sin\mu}{\cos\gamma}\right) \\
&\dot{\lambda}_\gamma = \lambda_x V\sin\gamma\cos\chi + \lambda_y V\sin\gamma\sin\chi - \lambda_h V\cos\gamma) \\
&\quad + \lambda_V V\cos\gamma - \lambda_\gamma\frac{g\sin\gamma}{V} \\
&\quad - \lambda_\chi\left(\frac{T\sin\alpha}{V} + kVC_L\right)\frac{\sin\mu\sin\gamma}{\cos^2\gamma} \\
&\dot{\lambda}_\chi = \lambda_x V\cos\gamma\sin\chi - \lambda_y V\cos\gamma\cos\chi \\
&\dot{\lambda}_\mu = a_\mu\lambda_\mu + \left(\frac{T\sin\alpha}{V} + kVC_L\right)\left(\lambda_\gamma\sin\mu - \lambda_\chi\frac{\cos\mu}{\cos\gamma}\right) \\
&\dot{\lambda}_T = a_T\lambda_T - \lambda_V\cos\alpha - \frac{\sin\alpha}{V}\left(\lambda_\gamma\cos\mu + \lambda_\chi\frac{\sin\mu}{\cos\gamma}\right) \\
&\dot{\lambda}_\alpha = a_\alpha\lambda_\alpha + \lambda_V(T\sin\alpha + kV^2C_{D_\alpha}) - \left(\frac{T\cos\alpha}{V} + kVC_{L_\alpha}\right) \\
&\quad \left(\lambda_\gamma\cos\mu + \lambda_\chi\frac{\sin\mu}{\cos\gamma}\right)
\end{aligned}
\tag{18}
$$

The boundary conditions for the co-states are given by

$$
\begin{aligned}
&\lambda_x(t_f) = \lambda_y(t_f) = \lambda_h(t_f) = \lambda_V(t_f) = \lambda_T(t_f) = \lambda_\alpha(t_f) = 0, \\
&\lambda_\gamma(t_f) = 2\gamma(t_f), \quad H(t_f) = 0
\end{aligned}
\tag{19}
$$

The optimum control inputs are found using Pontryagin's minimum principle:

$$
\begin{aligned}
&T_c = \frac{\lambda_T a_t}{\eta_T}, \quad \mu_c = -\operatorname{sign}(\lambda_\mu)\mu_{\max} \\
&\alpha_c = -\operatorname{sign}(\lambda_\alpha)\,\alpha_{\max}
\end{aligned}
\tag{20}
$$

From (20), we expect α_c and μ_c to follow a "bang–bang" profile during the ATA. The switching times, however, are difficult to estimate analytically. Therefore, we solve the optimal control problem numerically using GPOPSII (software available online at http://www.gpops2.com) which uses a direct optimization method. Results for the two cases $[\eta_x, \eta_y, \eta_h] = [1, 5, 1]$ and $[1, 1, 5]$ are plotted in Figure 8. These cases capture short and narrow volumes, respectively. In both cases, the aircraft performs a 3D turn. The angle of attack reaches the maximum value rapidly. The specific thrust is more or less constant, around 5 m/s^2. Although the maximum value of $\mu_c = 60°$ is attained in both cases, the important distinction is the instant at which the roll commences, with respect to the pull-up (which measures the time delay between the pull up to $\alpha_{\max}$ and the roll to $\mu_{c,\max}$). Note that, in both cases, μ returns to zero after the aircraft has turned through 140°. It is also worth noting that the duration of the maneuver is almost the same, approximately 2 s, in both cases.

The pull up to $\alpha_{\max}$ with wings more or less level (i.e. $\mu_c = 0$) causes the aircraft to climb and slow down. A larger time delay between the pull-up to $\alpha_{\max}$ and the roll to $\mu_{c,\max}$ causes the aircraft to slow down considerably while gaining altitude, after which it changes the heading rapidly before accelerating and descending to its previous altitude. On the other hand, a smaller time delay between the pull-up and the roll leads to a more or less steady turn, as is evident from Figure 8.

The above analysis leads to the hypothesis that aggressive turns can be performed by commanding constant values of T_c and $\alpha_c (= \alpha_{\max})$, while μ_c follows a three-segment "bang–bang" profile, as illustrated in Figure 9.

- Segment 1: $\mu_c = 0$ for $0 \le t \le \tau_d$, where the value of τ_d depends on the shape of the turning volume.
- Segment 2: $\mu_c = \pm\mu_{c,\max}$, while $\pi - |\chi - \chi_{\text{initial}}| > \Delta\chi_{\text{crit}}$, where $\Delta\chi_{\text{crit}}$ is the heading angle through which the aircraft turns while rolling from $|\mu| = \mu_{c,\max}$ to $\mu = 0$. We estimate $\Delta\chi_{\text{crit}}$ analytically later in this section.
- Segment 3: $\mu_c = 0$ as the aircraft recovers to level flight after a 180° heading change.

The physical interpretation of the three segments is as follows. In the first segment of Figure 9, the aircraft decelerates rapidly and attains a positive value of the flight path angle γ. There is, however, a trade-off involved here: the reducing flight speed tends to reduce the turn rate for a given combination of α and μ, while increasing γ increases the turn rate. Moreover, as the aircraft climbs for the duration of the first segment, it is to be expected that the height of the turning volume limits the duration of the first segment.

In the second segment, the aircraft banks to the maximum possible bank angle to achieve the largest possible turn rate and thereby the smallest possible turn radius (measured in the horizontal plane). In the third segment, the aircraft merely recovers to level flight.

The complete ATA maneuver primitive is described in Algorithm 1. The shape of the volume available for turning (narrow versus short) can be sensed and the duration of the first segment, τ_d, can be chosen accordingly. In a practical setting, τ_d corresponds to the time lapsed between the transmission of the pull-up and roll commands. Note that the value of τ_d is bounded from above by the time required for the aircraft to decelerate to the stalling speed, i.e. the speed below which the lift is insufficient to balance the weight of the aircraft.

Figure 10 depicts ATA trajectories for various values of τ_d, with $\mu_{\max} = 1.1$ rad. It is evident that a large value of τ_d

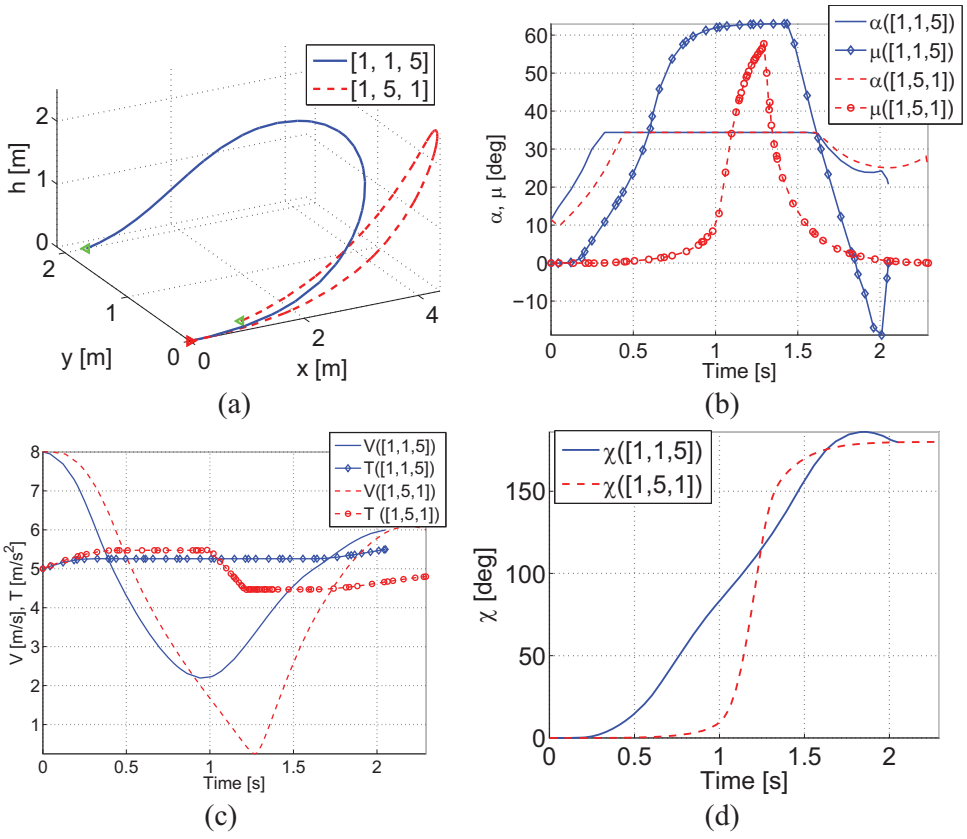

Fig. 8. Trajectory and flight parameters for two sets of $[\eta_x, \eta_y, \eta_h]$: [1, 1, 5] and [1, 5, 1] : (a) 3D trajectory; (b) wind axis angles; (c) thrust and speed; (d) heading angle.

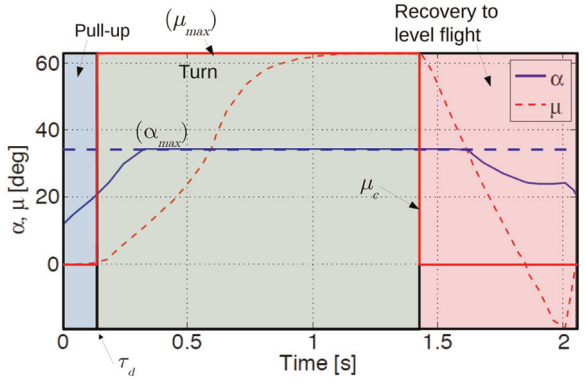

Fig. 9. An annotated version of Figure 8(b) showing the three stages of an ATA.

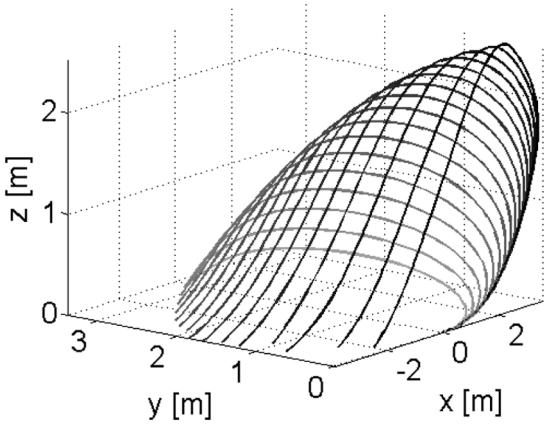

Fig. 10. Plots showing the aggressive turn trajectory for $\tau_d \in [0, 1]$, with the darker curves denoting a larger time delay.

permits turns inside a narrow volume, while smaller values lead to wider turns with a smaller change in altitude. This observation can be explained as follows. When the aircraft pulls up to $\alpha_{\max}$, it climbs rapidly and decelerates in the process. The increase in altitude as well as the reduction in speed are directly proportional to the duration of the pull-up. The increased flight path angle helps reduce the turn radius. The conclusions obtained from Figure 10 match those from Figure 8. The trajectories in Figure 8(a) have a profile similar to Figure 10.

Although the qualitative trends in Figure 10 are independent of the initial conditions, a non-zero γ can improve the turning performance significantly. A lower initial speed reduces the forward distance covered during the turn. However, it has virtually no bearing on the actual turn radius.

For the ATA maneuver in Algorithm 1, we need to estimate $\Delta\chi_{\text{crit}}$, which is the angle through which the aircraft

turns before commencing recovery to level flight. The value of $\Delta\chi_{\text{crit}}$ depends on the agility; when the agility is infinite, we would set $\Delta\chi_{\text{crit}} = 0$. For a robotic aircraft whose agility is finite, i.e. with $\dot{\mu} = a_\mu(\mu_c - \mu)$ as in (3), and with a_μ finite, we calculate $\Delta\chi_{\text{crit}}$ by assuming that V, γ and α do not change significantly in the short time $1/a_\mu$. Assuming that the robotic aircraft turns at a bank angle $\mu_{\max}$, we get the following expression for $\Delta\chi_{\text{crit}}$ using (2):

$$\Delta\chi_{\text{crit}} = \frac{T\sin\alpha/V + kVC_L}{a_\mu\cos\gamma}\sin\mu_{\max}$$

An approximate value of $\Delta\chi_{\text{crit}}$ can be found by setting $\mu_{\max} \approx \pi/2$ and $\cos\gamma \approx 1$:

$$\Delta\chi_{\text{crit}} \approx \frac{T\sin\alpha/V + kVC_L}{a_\mu}$$

Note that C_L is set at the outset by fixing α; k and a_μ are known parameters, and V can be measured. It may be possible to compute $\Delta\chi_{\text{crit}}$ continuously, and commence recovery when its value matches the remaining value of the heading change.

For the robotic aircraft considered here, $k = 0.37$, $C_L = 1.6$ during the turn, $a_\mu = 8\ \text{s}^{-1}$, while $V \approx 5$ m/s during the recovery phase. The thrust was set to $T \approx 5$ for the simulations in Figure 8. This gives $\Delta\chi_{\text{crit}} \approx 42°$, which is quite close to that obtained in Figure 8.

To compute the thrust T_c for the maneuver in Algorithm 1, we start by assuming a zero change in altitude and final speed, in which case energy balance implies that the role of thrust is to compensate for the energy dissipated due to the drag, so that

$$T_c = \frac{\int_0^{l_f} kC_D V^2\, dl}{\int_0^{l_f} dl} = \frac{kC_D\int_0^{t_f} V^3\, dt}{\int_0^{t_f} V\, dt} \tag{21}$$

where l_f denotes the length of the path flown by the robotic aircraft during the ATA maneuver. If we assume that the aircraft slows down almost to zero and the values of acceleration and deceleration are constant (i.e. $dt = dV/\text{acceleration}$, which allows us to replace dt in (21) by dV), we get

$$T_c = \frac{kC_D(\alpha_{\max})V_0^2}{2} \tag{22}$$

This value, however, needs to be used with caution because the aircraft need not recover all of its kinetic energy at the end of the turn (as seen in Figure 8). This can reduce the thrust requirement significantly as seen in Figure 8.

5. Implementation as part of a motion planning algorithm

The primary objective of this section is to show how the motion primitives derived in Sections 3 and 4 can be used in a motion planning algorithm for high-speed flight in a densely crowded environment. The motion planning algorithm derived in this section combines the aforementioned motion primitives with a RHC framework. The objective of the motion planning algorithm is to take the robotic aircraft to within a threshold distance of the goal. The threshold distance can be set of zero if the goal is a terminal destination, while a non-zero value can be chosen for the threshold distance if the goal is an entity to be observed or tracked.

Algorithm 1. ATA maneuver.

Result: $\chi \leftarrow \chi \pm \pi$
Initialize $t \leftarrow t_0$ and $\chi_f = \chi \pm \pi$
while $\chi \neq \chi_f$ **do**
  $\alpha_c = \alpha_{\max}$, T_c from (22)
  if $t > t_0 + \tau_d$ and $|\chi_f - \chi| > \Delta\chi_{\text{crit}}$ **then**
    $\mu_c \leftarrow \mu_{c,\max}$
  else
    $\mu_c = 0$
  end
end

5.1. RHC-based motion planning algorithm

The objective of the motion planner is to guide the robotic aircraft to the goal $(x_{\text{goal}}, y_{\text{goal}}, h_{\text{goal}})$. Let $\xi_0 = (x_0, y_0, h_0)$ denote the location of the robotic aircraft at an instant where the control inputs are to be computed. We choose points $\xi_i = (x_i, y_i, h_i) \in \mathcal{V}$ randomly, where $\mathcal{V}$ denotes the visible region and the index i satisfies $1 \le i \le N$ for a suitably large sample size N.

Let $\Sigma = \{\sigma_i(t)\}$ denote the set of trajectories $\sigma_i(t)$ which connect the starting point ξ_0 to the waypoint ξ_i, as shown in Figure 11. The mapping $\xi_i \mapsto \sigma_i(t)$ is obtained from the analytical formulae derived in Section 3, which, in fact, yield the map $\xi_i \mapsto \mathbf{u}_{c,i}$, the vector of constant control inputs (see Equations (11), (12), and (15)) which are required to fly the trajectory σ_i.

Let $\mathcal{T} = \{\mathcal{T}_j\} \subset \mathcal{V}$ denote the set of obstacles (each of which carries a unique index j). Let us denote the distance of an obstacle from a trajectory by $d(\sigma_i(t), \mathcal{T}_j)$. Let J denote the cost function whose value at a point ξ is denoted by $J(\xi, \xi_0)$. From the set of candidate way points $\{\xi_i\}$ $(1 \le i \le N)$, the motion planner chooses a point ξ_{next}

$$\xi_{\text{next}} = \arg\min_{\xi_i} \{J(\xi_i, \xi_0) \mid \min_j\{d(\sigma_i(t), \mathcal{T}_j)\} > \text{threshold};\ \mathbf{u}_{c,i}\ \text{admissible}\} \tag{23}$$

If none of the points $\xi_1, \ldots, \xi_N$ are found to be feasible, then the motion planner commands the ATA maneuver described in Algorithm 1 (see also Section 4).

In order to prevent overly conservative thresholds, one may allow the threshold to depend on the dynamics and a

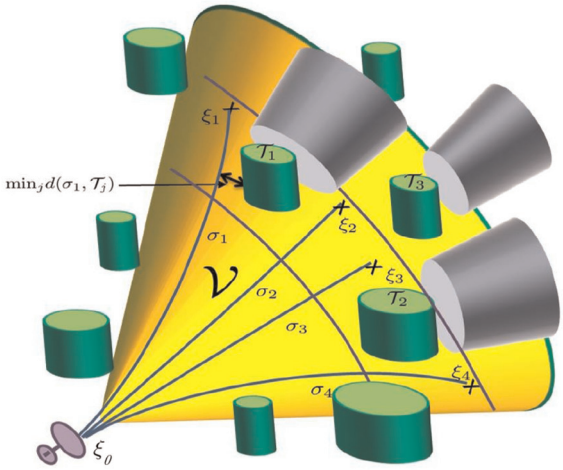

Fig. 11. The 3D sensing cone with the obstacles (labeled as $\mathcal{T}_i$), the candidate waypoints (marked by **X** and labeled by ξ_i), and the trajectories to the waypoints (marked by σ_i). The grey areas are occluded and hence not sampled for candidate waypoints.

stochastic model of the disturbances (Hu et al., 1999). The threshold distance can be calculated using information about the convergence rate and the robustness of the flight controller, as illustrated presently. An alternate but related approach is to use the expected deviation from the nominal trajectory in the design of the nominal primitive itself (Schouwenaars et al., 2004).

One interesting point in connection with choosing the threshold distance is avoiding local trapping regions. For example, due to a limited sensing range, the robot could fly into a funnel which, in some cases, may lead to an enclosed region inside which an ATA may not be safely executed. To prevent such events, the threshold should be chosen to be large enough to accommodate an ATA maneuver. The threshold may be chosen as a function of the flight speed, and it can be made time-varying if required.

5.1.1. Computation of deviation about mean trajectory. We can find a conservative estimate for the threshold distance by letting $\cos\gamma \approx 1$ so that the translational motion can be approximated by

$$\dot{x} = V\cos\chi, \quad \dot{y} = V\sin\chi, \quad \dot{\chi} = kVC_L\sin\mu$$

and hence (since $d\chi = kVC_L \sin\mu\, dt$)

$$\frac{dx}{d\chi} = \frac{\cos\chi}{kC_L\sin\mu}, \quad \frac{dy}{d\chi} = \frac{\sin\chi}{kC_L\sin\mu} \tag{24}$$

Ideally, the aircraft would turn with a constant value of $\mu = \mu_c$ and $C_L = C_{L_c}$. If the values of μ and C_L differ from μ_c and C_{L_c}, respectively, we get errors $e_x = x - x_c$ and $e_y = y - y_c$, where $(x_c(t), y_c(t))$ is the trajectory obtained using the commanded inputs. From (24), the error dynamics are given by

Algorithm 2. Motion planner for agile flight.

Result Safe, fast flight through a forest and arrival at $\xi_{\text{goal}} = (x_{\text{goal}}, y_{\text{goal}}, h_{\text{goal}})$ initialization: fly = 1
// fly = 1: routine flight (Section 3); fly = 0: ATA (Section 4)
Position: $\xi_0 = (x_0, y_0, h_0)$
while $d_{\text{goal}} > d_{\text{nom}}$ **do**
  Draw N samples $\mathcal{V}_n := (x_i, y_i, h_i) \in \mathcal{V}, 1 \le i \le N$
  Define out = zeros(n, 5) (cost, feasibility, control)
  for every $\xi_i \in \mathcal{V}_n$ **do**
    Compute trajectory $\sigma_i(t)$ and control inputs $\mathbf{u}_{c,i}$ from (11), (12) and (15)
    Compute $d(\sigma_i(t), \mathcal{T}_j) \forall\, \mathcal{T}_j \in \mathcal{V}$
    if $\exists i$ s.t. min $(d(\sigma_i(t), \mathcal{T}_j))$>threshold **then**
      out $(i,:) = [J(\xi_i, \xi_0), 1, \mathbf{u}_{c,i}]$
    end
  end
  if max (out(:, 2)) = 0 **then**
    // No feasible trajectory
    fly = 0
  else
    find $j = \arg\min_i\{\text{out}(i, 1) | \text{out}(i, 2) = 1\}$
    $\mathbf{u}_c \leftarrow \mathbf{u}_{c,j}$ = out (j, 3: 5)
    set time of flight $t_{\text{flight}} = \sigma_j^{-1}(\xi_j)$
  end
  if fly = 1 **then**
    Fly "routinely" with control $\mathbf{u}_c$ for t_{flight}
    Update position ξ_0
    recompute d_{goal}
  else
    Perform ATA using Algorithm 1
    set fly = 1
    Update position ξ_0
    Recompute d_{goal}
  end
end

$$\begin{aligned} \frac{de_x}{d\chi} &= \frac{\cos\chi}{kC_{L_c}\sin\mu_c}\left(\frac{C_{L_c}\sin\mu_c}{C_L\sin\mu} - 1\right) \\ \frac{de_y}{d\chi} &= \frac{\sin\chi}{kC_{L_c}\sin\mu_c}\left(\frac{C_{L_c}\sin\mu_c}{C_L\sin\mu} - 1\right) \end{aligned} \tag{25}$$

so that

$$\begin{aligned} \sqrt{e_x^2 + e_y^2} &\le \frac{1}{kC_{L_c}\sin\mu_c}\left\|\frac{C_{L_c}\sin\mu_c}{C_L\sin\mu} - 1\right\|_\infty \sqrt{2 - 2\cos\chi} \\ &\approx \frac{1}{kC_{L_c}\sin\mu_c}\left\|\frac{C_{L_c}\sin\mu_c}{C_L\sin\mu} - 1\right\|_\infty \chi \end{aligned} \tag{26}$$

where $\|\cdot\|_\infty = \sup_t|\cdot|$. The angle χ will eventually depend upon how long the aircraft turns.

Note that $1/(kC_L \sin\mu_c) = R_c$, the commanded turn radius. If we assume a 5% error in $C_L \sin\mu$ with respect to the commanded value, then $\left\|\frac{C_{L_c}\sin\mu_c}{C_L\sin\mu} - 1\right\|_\infty < 0.05$. We can let $\chi \approx 1$, i.e. a turn through 60° before the next update, in which case the size of the tube is bounded above by $0.05R_c$, and 2 m is a reasonable estimate for the drift, assuming a turn radius of 40 m.

For the limiting case where $\mu_c = 0$, i.e. when the aircraft is required to fly straight, we can estimate the size of the tube assuming first-order convergence for μ, i.e. $\mu(t) = \mu_0 e^{-a_\mu t}$, where $a_\mu > 0$ is the time constant for the μ dynamics and μ_0 is the initial error in μ (after the agility has been accounted for). Since the value of χ and μ are small, we estimate the drift from a straight line path (given, without loss of generality, by $\chi = 0$) using

$$\begin{aligned} \ddot{y}(t) \approx kV^2 C_L \mu = g\mu_0 e^{-a_\mu t} \Rightarrow |y(t)| \\ \leq \frac{g}{{a_\mu}^2} |\mu_0| \left| a_\mu t + e^{-a_\mu t} - 1 \right| \end{aligned} \tag{27}$$

where $y(0) = 0$ and $\dot{y}(0) = 0$ are assumed.

The above equation allows us to choose the update time t as a function of the agility a_μ, the permissible value of deviation and expected initial error $|\mu_0|$. In general, a large value of a_μ (i.e. a higher amount of agility) permits a larger sampling time. Interestingly enough, the permissible value of sampling interval is independent of the flight speed.

5.1.2. Motion planning. The motion planner first runs at the instant of commencing flight. Thereafter, it runs at the end of pre-defined interval, or after an ATA maneuver if one needs to be performed. The motion planner stops when the robotic aircraft's distance from the goal, $d_{\text{goal}} \triangleq \sqrt{(x_{\text{goal}} - x_0)^2 + (y_{\text{goal}} - y_0)^2 + (h_{\text{goal}} - h_0)^2}$, is less than or equal to some nominal value d_{nom}. The complete algorithm has been described in Algorithm 2.

Note that the sampling instances, i.e. instances when the control inputs are computed and commanded, need not coincide with proximity to the waypoints. In fact, sampling should be performed much before the robotic aircraft reaches the intended waypoint, in order to steer away from obstacles that may have initially been beyond its sensing radius. This approach lends the motion planning algorithm an RHC-like structure (Morgan et al., 2014).

The RHC-based algorithm described above has no built-in provision to prevent cyclic paths from recurring. The only way to do as much is to preserve a memory of the path chosen, as well as a record of paths *not taken*, i.e. by combining map-building and navigation (Choset, 1996; Oriolo et al., 1998; Choset, 2001). A coverage-based module can be readily incorporated into Algorithm 2.

A final point concerns the computational cost of the motion planning algorithm at each running instant. Once the N candidate waypoints are chosen, the motion primitive takes $\mathcal{O}(N)$ steps for computation, since it is in the form of algebraic expressions, together with the computation of the center of the circular arcs which constitute the trajectory to each waypoint. The distance between a given tree and the trajectory can be computed using another algebraic expression d (centre of arc, tree) $- R$, where R is the radius of the circular arc. Therefore, the computational time is $\mathcal{O}(N) \times \mathcal{O}(\text{card}(\mathcal{T}))$, where $\text{card}(\mathcal{T})$ denotes the cardinality of the set of trees.

5.2. Simulations

We demonstrate the capabilities of the motion planner described in Algorithm 2 and the ATA maneuver through simulations performed in Matlab. The equations of motion presented in Appendix B are used for simulation, unlike the restricted set (2) used to derive the motion primitives. The robotic aircraft weighs 100 g, has a wing area $S = 0.5$ m^2, while the coefficients of lift and drag are given by $C_L = 0.3 + 2.5\alpha$, and $C_D = 0.03 + 0.3\,C_L^2$. The maximum values of thrust and wind axis roll angle are given by $T_{\max} = 1.2$ and $\mu_{\max} = \pm 1.1$ rad. The limiting angle of attack (used as a proxy for the stall angle of attack) is $\alpha_{\max} = 35°$. A forest with a specified number of trees is generated such that the coordinates of the trees and their radii are chosen through (mutually independent) Poisson distributions, and the tree radii are constrained between 0.5 and 1 m.

Let ξ_0 denote the position of the robotic aircraft at which sampling is performed, and for a candidate waypoint ξ_i, let $\theta_{\text{goal}}(\xi_0, \xi_i)$ denote the angle between the segment $\xi_i - \xi_0$ and $\xi_0 -$ goal, i.e. $\theta_{\text{goal}}(\xi_0, \xi_i)$ measures the bearing to the waypoint in relation to the bearing to the goal. Then, the "cost" of choosing ξ_i is defined by $J_i(\xi_i, \xi_0) = (1 - \cos(\theta_{\text{goal}}(\xi_0, \xi_i)))$. This particular cost function is, by no means, unique or optimal in any sense. It can be replaced by a function of the user's choice.

The stopping condition is set to $d_{\text{goal}} < 20$ m, while the threshold distance in (23) is set to 2 m (see Section 5.1.1). The start point and the goal are at (20, 20) and (190, 190), respectively.

Figure 12 shows the simulation of an aircraft flying through a $200 \times 200\text{m}^2$ forest with 500 trees distributed randomly with a Poisson distribution. The commanded speed is set to 9 m/s, while in each instant of running the motion planner, a total of 100 points are sampled in the visible space as candidate waypoints. Figure 13 shows the zoomed in view of an area where a series of ATA maneuvers is employed to navigate a particularly dense patch.

The results show that the motion planning algorithm (Algorithm 2) successfully guides the aircraft through the forest. Interestingly, the ATA maneuver was required even at a slower flight speed of 6 m/s (a different case from the example shown in Figure 12), which demonstrates its importance during flight in obstacle-rich environments.

Figure 14 shows the mean statistics obtained by simulating flight at various speeds through a 200 m $\times$ 200 m forest with different number of trees. A surprising observation is that the expected time of flight from the starting point to the end point does not change much as the flight speed increases. This is due to an increase in the number of ATA maneuvers required at higher flight speeds. Moreover, at higher flight speeds, the proportion of failed flights increases; a flight is said to have failed if the aircraft is trapped in a large number of ATA loops. In simulations, the flights were deemed to have failed if the number of ATA maneuvers exceeded 20. The fail-safe speed is thus

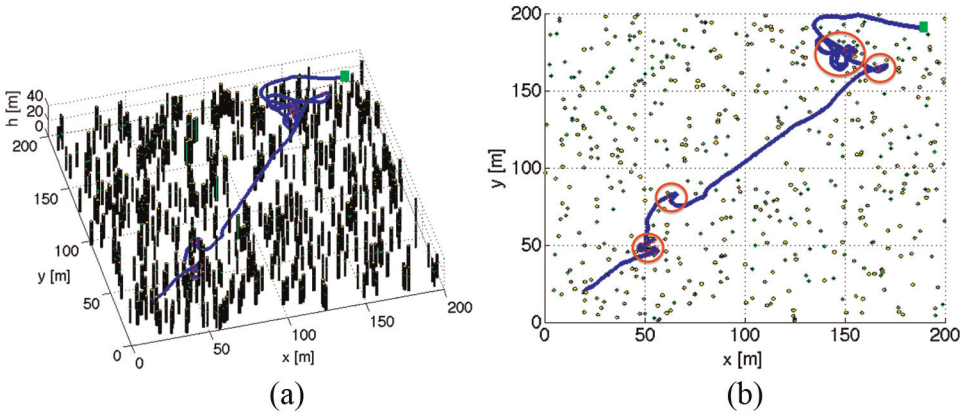

Fig. 12. (a) Trajectory of the robotic aircraft in 3D space as it navigates a forest and (b) its projection on the x–y plane. Red circles in the x–y projection denote the locations of the ATA maneuvers.

defined as the maximum speed at which no such trapping regions are found. Not surprisingly, the fail-safe speed rises rapidly as the density of trees in the forest decreases. It must also be noted, from Figure 14(b), that the average number of ATAs is around 1 as the flight speed reduces to 4 m /s. It is expected that ATA maneuvers will extend the speed envelope in a similar manner when used with other path planners as well.

6. Experiments

6.1. Demonstration of the motion primitives in Section 3

In order to demonstrate the motion primitives derived in Section 3, and the time-delay-based switching logic, we conducted a series of indoor flight tests using the Parkzone MiniVapor (shown in Figure 15(a) together with a photo of the testing volume, Figure 15(b)). The geometric and inertial properties of the MiniVapor are summarized in Table 8. The Vicon motion capture system was used to obtain the position and the orientation of the aircraft. The experiments were conducted inside a test volume measuring 7 m $\times$ 4 m $\times$ 2.5 m.

A notable feature of the MiniVapor is the absence of ailerons, and hence the lack of direct roll control capability. This naturally results in a small value of a_μ, which measures the roll agility of the aircraft (see Equation (2)). The aircraft has an moving vertical tail which provides combined roll–yaw control, which requires a critical modification in the implementation of the motion primitives. Moreover, for the experiments, we did not control the altitude of the aircraft owing to the short duration of the flight tests. Instead, the elevator and the thrust regulated the flight path angle γ as described presently.

The rate of change of the wind axis roll is given by

$$\dot{\mu} = \frac{p\cos\alpha + r\sin\alpha}{\cos\beta} \tag{28}$$

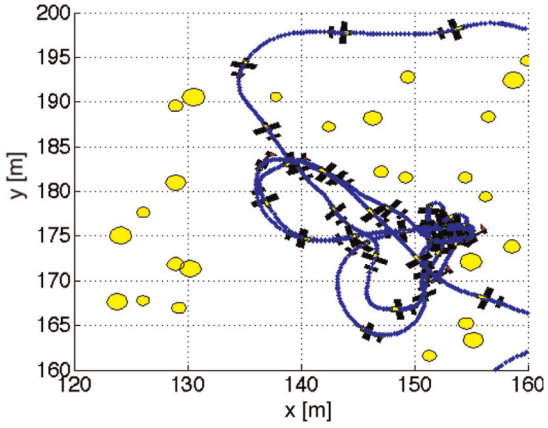

Fig. 13. Magnified view from Figure 12 showing the trajectory of the robotic aircraft as it performs a series of ATA maneuvers in a dense patch of trees in the forest.

Table 2. Key properties of the MiniVapor.

Property	Value (in SI units)
Mass	11 g (including tracking markers)
Wing span and chord	0.28 and 0.12
Principal moment of inertia ($\times 10^{-4}$)	0.33, 2.8, 3.13 (estimated)
Maximum thrust (N)	0.09
Maximum specific thrust (thrust/mass)	8.18
Non-dimensional constant $k = \rho S/2m$	1.86
Typical cruising speed range	2 − 4 m/s

Consider the problem of stabilizing the equilibrium condition $\mu = 0$ using μ feedback and the vertical tail (or rudder) input, δ_r, as the control input. The deflection δ_r gives rise to rolling as well as a yawing moment (L and N):

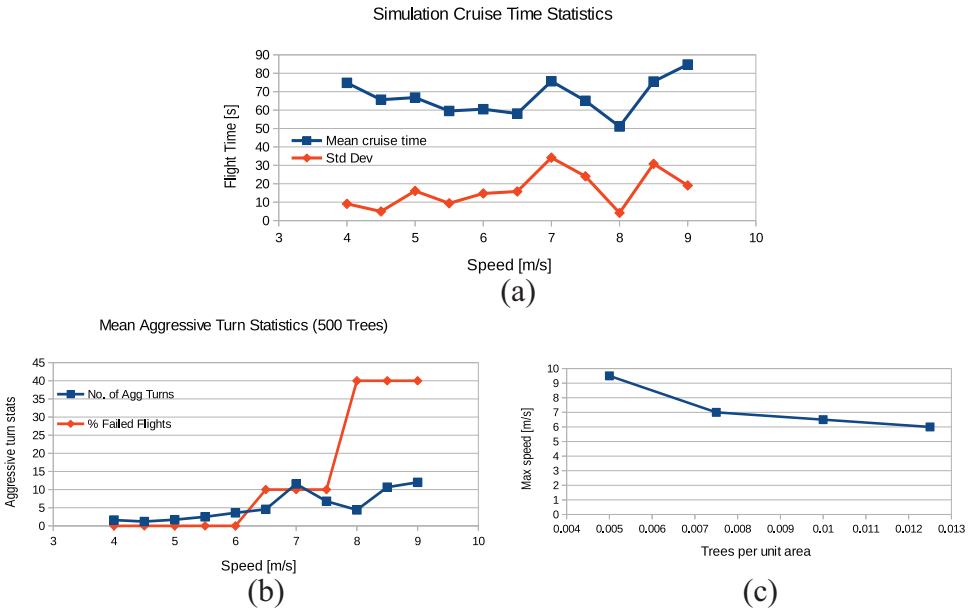

Fig. 14. Statistics of (a) cruise time and (b) aggressive turns for various flight speeds in a forest with 500 trees, and (c) the maximum fail-safe speed as a function of the tree density (number of trees per square meter).

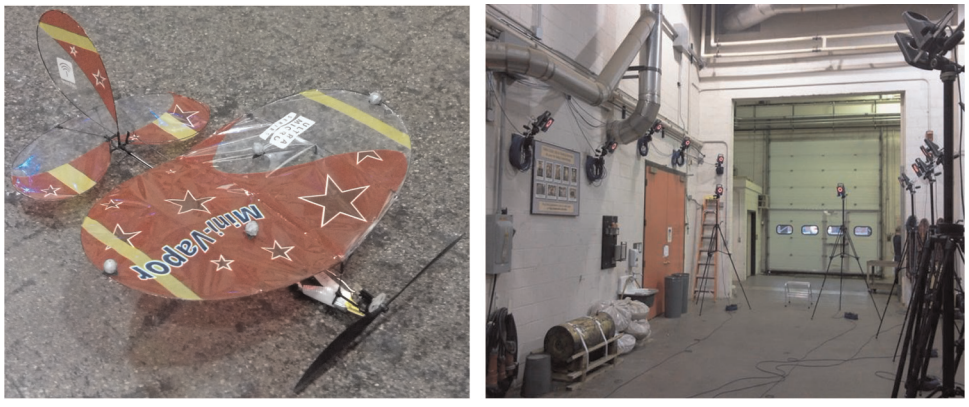

Fig. 15. The experimental setup: (a) an off-the-shelf MAV called the Parkzone MiniVapor and (b) the testing area with a Vicon motion capture system.

$$L(\delta_r) = L_{\delta_r}\delta_r,\ \ N(\delta_r) = N_{\delta_r}\delta_r$$

where $L_{\delta_r} < 0$ and $N_{\delta_r} > 0$ (see Appendix B). If the wind-axis roll angle is perturbed by a small $\Delta\mu > 0$ (without loss of generality), then in order to ensure that the incremental $\Delta\dot{\mu} < 0$, the rudder can try to make Δp and/or Δr negative by providing a negative rolling and/or negative yawing moment. However, these two effects cannot be achieved simultaneously due to the differing signs of L_{δ_r} and N_{δ_r}. In fact, μ-feedback leads to oscillations except when a very small control gain is used. However, a small gain slows down the stabilization to the point where it is practically useless.

A more appropriate control design method is to map the *steady-state value* of μ to a steady state value of r. We start with an expression derived in Appendix B:

$$r = (\cos\alpha\cos\gamma\cos\mu - \sin\alpha\sin\gamma)\dot{\chi}$$

We note that $\sin\gamma\sin\alpha \ll \cos\alpha\cos\gamma\cos\mu$ under routine flying conditions. Substitution for $\dot{\chi}$ from (2) gives

$$r = \cos\alpha\cos\mu\left(kVC_L + \frac{T\sin\alpha}{V}\right)\sin\mu$$

Finally, by substituting for $\dot{\gamma} = 0$ in (2), we get

$$r = \frac{g}{V}\cos\alpha\cos\gamma\sin\mu$$

This is a bijective mapping between steady-state values of μ and r, which allows us to map μ_c, which is commanded by the motion primitive, to a commanded value r_c for control design:

$$r_c = \frac{g}{V}\cos\alpha_c\sin\mu_c$$

with μ_c obtained from (15), and with the additional assumption $\cos\gamma \approx 1$. Although this is certainly not the case for the ATA maneuver, designing the ATA maneuver with a saturated rudder deflection eliminates any necessity to use the above mapping for ATA. It remains to explain the selection of α_c and T_c. For experiments, we set $\gamma_c = 0$ (thereby not requiring any particular altitude). The flight speed command, denoted be V, is mapped directly to the elevator deflection δ_e using an empirical formula obtained from flight tests:

$$\delta_{e_1} = 0.1846 - \frac{0.78}{V^2\cos\mu_c} \tag{29}$$

where the additional subscript '1' is used to denote a feed-forward signal, pending the addition of some feedback terms. The choice of a feed-forward $V - \delta_e$ map, as against a feedback controller, is justified by the well-damped pitching dynamics. The expression for α_c is determined in a similar fashion:

$$\alpha_c = 0.12 + \frac{0.981}{V^2\cos\mu_c}$$

Note that the choice of μ_c from (15) is independent of α_c and depends only on V and the location of the waypoint.

The elevator deflection command is the sum of δ_{e_1} from (29) and terms that compensate for γ:

$$\delta_e = \delta_{e_1} + 0.4\gamma + 0.7\int_0^t \gamma\,dt$$

The γ-terms are necessary to prevent the aircraft from losing altitude rapidly during turns. The thrust T_c is chosen as follows:

$$T_c = 0.8 - \gamma - 1.2\int_0^t \gamma\,dt$$

where the bias value of 0.8 is the minimum value of thrust required for level flight. The inner-loop controller for δ_r was a proportional-integral law:

$$\delta_r = 0.08(r_c - r) + 0.35\int_0^t (r_c - r)\,dt$$

In our description of the results, we will frequently refer to a point by its coordinates (x, y). Unless otherwise stated, both x and y are given in meters.

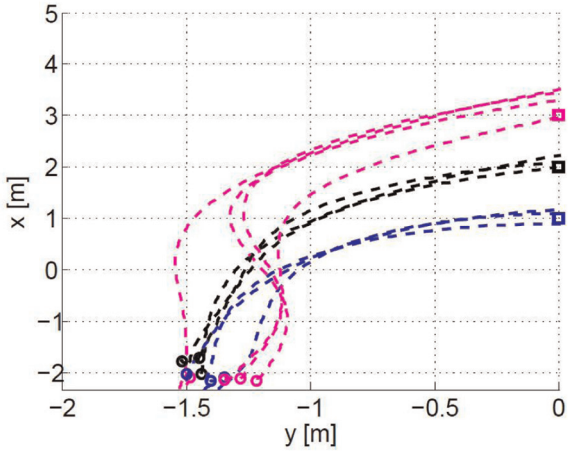

Fig. 16. Trajectory of the robotic aircraft during ten experiments involving flights to three different waypoints (denoted by □). The aircraft was hand-launched, which reflects in the discrepancy in the initial conditions.

Figure 16 shows the results of a series of flight tests where the objective was to fly to only one waypoint, after which the flight terminated. The aircraft was hand-launched from $(-2.2, -1.4)$. The calculation of the value μ_c (see Equation (15)) explicitly made use of the information about the launching point and assumed that the aircraft flew in a straight line parallel to $y = 0$ until it commenced a turn. The errors seen in Figure 16 arose primarily from this assumption. Nevertheless, the maximum miss distance was seen to be 45 cm in the results shown in Figure 16. In four out of five cases, the error in the final position averages around 10 cm.

Figure 17 shows the trajectory of the robotic aircraft during fifteen flight tests to the waypoint [5, 1]. The experiment was similar to that reported in Figure 16, except that the waypoint was commanded only after the aircraft crossed $x = 2$ m, until which point the controller was given a command of $\mu_c = 0$. Upon activation at $x = 2$ m, the controller used a time delay of $\tau_a = 0.8$ s to compute the drift trajectory using the instantaneous values of the flight parameters, following the approach in Section 3.2. The maximum miss distance is 75 cm, and that too during only one of the flights. In all other cases, the maximum error distance is less than 50 cm, and the mean error across the 15 flights was 20 cm. Figure 18 is a montage showing flight from the hand-launch to the desired waypoint.

Figure 19 shows the trajectory of the robotic aircraft as it flew a path with two waypoints in addition to a starting point located around $[-2, -1]$. The first waypoint was at [2, 0], while the second waypoint [5, 1] became the commanded waypoint after the aircraft crossed $x = 2$. Unlike the two flight tests described earlier (Figures 16 and 17), the control law was switched on as soon as the aircraft attained a minimum speed of 1.5 m/s. The transient behavior of the aircraft in the initial moments of flight introduced a significant variation in flight paths across the five

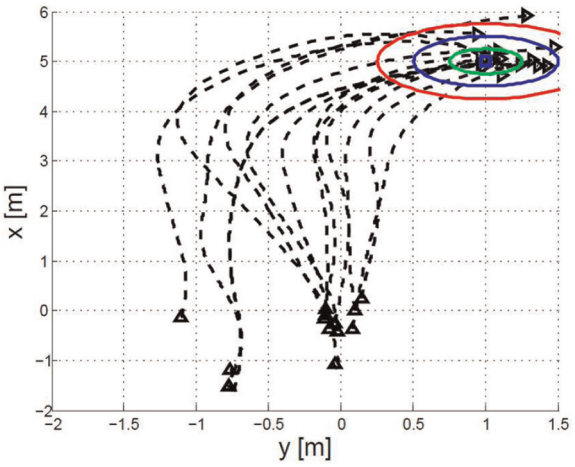

Fig. 17. Trajectory of the robotic aircraft during 15 flights to the waypoint [5, 1] (denoted by □). The three ellipses are actually circles (distorted due to the axes) which correspond to error radii of 0.25, 0.5 and 0.75 m. Some of the landings can be seen in the supplementary video (Extension 1).

tests. The most significant effect of the variation is the 180° range for the final heading angles (χ) of the aircraft after crossing the waypoint [5, 1]. This is in stark contrast to Figures 16 and 17, where the aircraft crossed the commanded waypoint with more or less the same heading in each flight test. The results in Figure 19 also demonstrate the importance of allowing the transient behavior to attenuate before the primitives-based controller is activated.

6.2. Effect of control laws on ATA

In order to examine the effect of including an inner-loop controller on the performance of an ATA, we implemented a controller of the form (Paranjape et al., 2013c)

$$\begin{aligned} p_c &= k_{p,\mu}(\mu_c - \mu) + k_{I,\mu}\int_0^t (\mu_c - \mu)\,dt \\ \delta_r &= k_{p,p}(p_c - p) + k_{I,p}\int_0^t (p_c - p)\,dt \end{aligned} \tag{30}$$

Note that a roll rate-based controller is quite effective when turning rapidly, unlike while trying to stabilize around level flight. This control law was tested experimentally on the MiniVapor (see Figure 15(a)). Some of the flight tests are included in the supplementary video, Extension 1. Figure 20 shows the turn radius as a function of the time delay τ_d between the pull-up and roll. Simulation results are shown alongside the experimental data.

The simulations show that, unlike the conclusions of Figure 10, the turn radius is optimized for a zero time delay, which is also verified in experiments. The variation in turn diameters is about 0.1 m (10 cm) for the complete range of τ_d considered here, which also contrasts sharply with Figure 10. The discrepancy between the simulation and experiments, in Figure 20, is largely due to the modeling uncertainties, particularly in the moments of inertia and drag estimates.

7. Inverse-design principles

We present some inverse-design principles for robotic aircraft intended for high-speed flight through densely crowded spaces. First, Equations (2) and (6) yield the following expression for the turning radius R after ignoring the contribution from T: $R = \frac{\cos^2\gamma}{kC_L \sin\mu}$,[1] which provides the upper bound on the turn radius for given C_L and k. The expression for R is independent of V: therefore, the turn radius, which is a measure of how crowded an obstacle field can be flown through, is *independent of the flight speed*, but depends strongly on k (which is a scaled inverse of the wing loading) and the maximum achievable value of C_L, which depends primarily on the Reynolds number and the shape of the airfoil.

Second, a key metric which constrains the class of navigable obstacle fields is a_μ, which measures how rapidly an aircraft can change its turn rate. It turns out that $a_\mu \propto V^2$, i.e. the aircraft agility *increases* with its speed. Therefore, high-speed flight is a better alternative to low speed flight even from the point of view of flight dynamics, aside from making for a spectacular sight.

Finally, we note that the turn radius is inversely proportional to $k = \frac{\rho S}{2m}$, which is clearly an important design parameter. A large value of k yields a tighter ATA maneuver, and the volume inside which an ATA maneuver can be performed increases rapidly with reducing k. However, a larger value of k is ideally suitable for *slow* flight. Therefore, k

Fig. 18. Snapshot showing flight to the waypoint (flanked by the two tripods) following a hand-launch.

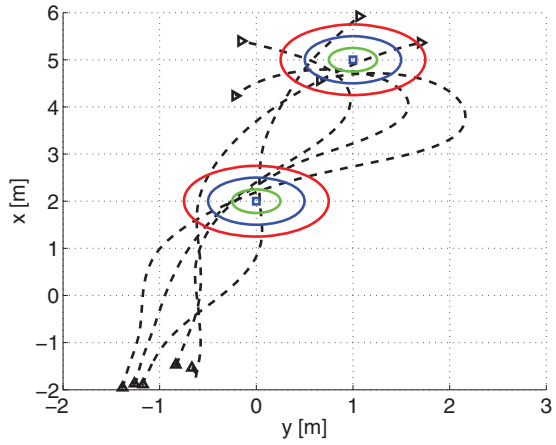

Fig. 19. Trajectory of the robotic aircraft during five flights across two waypoints. In all cases, the aircraft was commanded to fly to [2, 0] and then to [5, 1], with the crossing of $x = 2$ triggering the change in waypoint command. The three ellipses are circles, as in Figure 17, correspond to error radii of 0.25, 0.5 and 0.75 m.

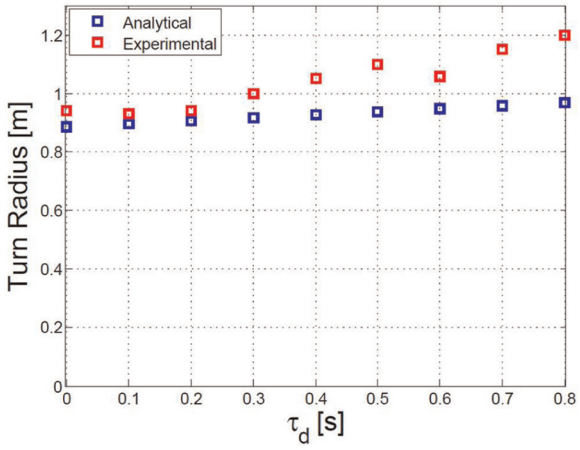

Fig. 20. Simulation results and experimental data showing the effect of a control law (Paranjape et al., 2013c) on the turn diameter during an ATA. Note that the turn radius is measured in the horizontal (x–y) plane.

needs to be optimized for the class of obstacle fields that the robotic aircraft is designed to cross as well as the desired time of crossing.

Suppose that the obstacle field density requires a minimum value of R, denoted as $R_{\min}$. The average value of $\cos^2\gamma \approx 1$ while $C_{L,\max} = 1$ is also a reasonable estimate. Typical values of μ are in excess of 30° (load factor of 1.2 during a level turn), and we may thus set $\sin\mu = 0.5$. Then, we need k to satisfy

$$\frac{\rho S}{2m} \triangleq k \geq \frac{2}{R_{\min}}, \text{ i.e., } \frac{m}{S} \leq \frac{R_{\min}}{3}$$

The above expression automatically imposes an upper bound on the wing loading. The bound can be relaxed by computing $\cos\gamma$ more accurately, but that process requires a design to start with. Thus, it is possible to perform inverse design iteratively. An increased mass allowance can be used for installing improved sensing and computational capability on board. As a numerical illustration, $\gamma = 20°$ permits an additional mass equal to 12% of the first estimate (which was obtained by assuming $\gamma = 0$).

8. Conclusion

In this paper, we presented a novel approach for constructing and stitching together two families of motion primitives for agile autonomous robotic aircraft flying at high speeds in dense obstacle fields. The motion primitive are continuously parametrized in the space of the control inputs. The key idea behind the stitching logic for primitives is to model the transition between two maneuver states as an instantaneous switch between the two states following a characteristic time delay which depends on the agility of the robot.

The first family of primitives consists of *steady* 3D turns between successive waypoints chosen by the motion planning algorithm. By assuming steadiness, we derived algebraic formulae between the control inputs required to accomplish the turn, and the distance and the bearing between successive waypoints. The stitching between successive primitives was modeled as an instantaneous switch between the two maneuvers following a characteristic time delay which depends on the roll agility of the aircraft. The resulting stitching logic was directly incorporated into the derivation of the algebraic expressions for the control inputs.

The second family consists of ATA primitives, designed to allow aircraft, flying at high speeds, to back-track from impenetrable regions of the obstacle field by performing *instantaneous* turns inside a small volume. The instantaneous turns are composed of three segments. In particular, the duration of the first segment is equal to the time delay between the command to pull-up to the prescribed angle of attack and the command to roll to the maximum possible angle for the turn. This time delay can be optimized for the shape of the turning volume. However, the quantitative relation between the shape of the turning volume and the time delay was shown to be sensitive to the inner-loop controllers.

Both families of primitives were demonstrated in experiments to prove their feasibility as well as to demonstrate the effect of inner-loop controllers on the primitives. The primitives can be combined readily with an off-the-shelf motion planning algorithm. This was demonstrated by numerically simulating an RHC-based motion planning algorithm which directly incorporated the motion primitives derived in the paper. The simulations also demonstrated the maximum speed at which high speed is flight is possible without getting trapped locally in ATA loops.

Acknowledgements

The authors gratefully acknowledge the contributions of Sunil Patel to the experiments reported in this paper. The authors would also like to thank the reviewers for their thorough and critical reviews which helped improve the manuscript to its present form.

Funding

This work was supported by the ONR (grant number N00014-11-1-0088) and the NSF (grant number IIS-1253758).

Notes

1. This is a well-known expression for the turning radius and is also found in elementary textbooks on flight mechanics (Phillips, 2009).

References

Berg-Taylor K, Seo K and Chung SJ (2008) Sensor based path planning in highly constrained environments for agile autonomous vehicles. In: *AIAA guidance, navigation, and control Conference*, Honolulu, HI, USA.

Bry A, Bachrach A and Roy N (2012) State estimation for aggressive flight in GPS-denied environments using onboard sensing. In: *Proceedings IEEE international conference on robotics and automation (ICRA)*.

Celik K, Chung SJ, Clausman M and Somani AK (2009) Monocular vision SLAM for indoor aerial vehicles. In: *International conference on intelligent robots and systems (IROS)*, pp. 1566–1573.

Choset H (1996) *Sensor Based Motion Planning: the Hierarchical Generalized Voronoi Graph*. PhD Thesis, California Institute of Technology.

Choset H (2001) Coverage for robotics - a survey of recent results. *Annals of Mathematics and Artificial Intelligence* 31: 113–126.

Dani A, Panahandeh G, Chung SJ and Seth (2013) Image moments for higher-level feature based navigation. In: *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Tokyo, Japan, pp. 602–609.

Dever C, Mettler B, Feron E, Popovic J and McConley M (2006) Nonlinear trajectory generation for autonomous vehicles via parametrized maneuver classes. *Journal of Guidance, Control, and Dynamics* 29(2): 289–302.

Frazzoli E, Dahleh MA and Feron E (2002) Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control and Dynamics* 25(1): 116–129.

Frazzoli E, Dahleh MA and Feron E (2005) Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics* 21(6): 1077–1091.

Goerzen C, Kong Z and Mettler B (2010) A survey of motion planning algorithms from the perspective of autonomous UAV guidance. *Journal of Intelligent Robotic Systems* 57: 65–100.

Gray A, Gao Y, Lin T, Hedrick JK, Tseng HE and Borrelli F (2012) Predictive control for agile semi-autonomous ground vehicles using motion primitives. In: *American control conference*, Montreal, QC, Canada, pp. 4239–4244.

Hu J, Lygeros J, Prandini M and Sastry S (1999) Aircraft conflict prediction and resolution using Brownian motion. In: *Proceedings of the 38th conference on decision and control*, Phoenix, AZ.

Karaman S and Frazzoli E (2012) High-speed flight in an ergodic forest. In: *IEEE international conference on robotics and automation*.

Kavraki LE, Svestka P, Latombe JC and Overmars M (1996) Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4): 566–580.

Kehoe JJ, Watkins AS and Lind R (2006) A time-varying hybrid model for dynamic motion planning of an unmanned air vehicle. In: *AIAA guidance, navigation, and control conference*, Keystone, CO, USA.

Kelley HJ (1971) Reduced-order modeling in aircraft mission analysis. *AIAA Journal* 9(2): 349–350.

Kelley HJ and Edelbaum TN (1970) Energy climbs, energy turns and asymptotic expansions. *Journal of Aircraft* 7(1): 93–95.

Kuo BC and Golnaraghi MF (2003) *Automatic Control Systems*. New York: John Wiley & Sons.

Langelaan J and Rock S (2005) Towards autonomous UAV flight in forests. In: *AIAA guidance, navigation and control conference*, San Francisco, CA, USA.

LaValle SM (2006) *Planning Algorithms*. Cambridge: Cambridge University Press.

Matsumoto T, Konno A, Suzuki R, Oosedo A, Go K and Uchiyama M (2010) Agile turnaround using post-stall maneuvers for tail-sitter VTOL UAVs. In: *2010 IEEE/RSJ international conference on intelligent robots and systems*, Taipei, Taiwan, pp. 1612–1617.

Morgan D, Chung SJ and Hadaegh FY (2014) Model predictive control of swarms of spacecraft using sequential convex programming. *Journal of Guidance, Control, and Dynamics* 37(6): 1725–1740.

Ollero A and Heredia G (1995) Stability analysis of mobile robot path tracking. In: *Proceedings of IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Pittsburgh, PA, USA, pp. 461–466.

Oriolo G, Ulivi G and Vendittelli M (1998) Real-time map building and navigation for autonomous robots in unknown environments. *IEEE Transactions on Systems, Man and Cybernetics - Part B: Cybernetics* 28(3): 316–333.

Paranjape AA and Ananthkrishnan N (2006) Combat aircraft agility metrics - a review. *Journal of Aerospace Sciences and Technologies* 58(2): 1–16.

Paranjape AA, Chung SJ and Kim J (2013a) Novel dihedral-based control of flapping-wing aircraft with application to perching. *IEEE Transactions on Robotics* 29(5): 1071–1084.

Paranjape AA, Chung SJ and Selig MS (2011) Flight mechanics of a tailless articulated wing aircraft. *Bioinspiration and Biomimetics* 6(2): 026005.

Paranjape AA and Meier KC, Shi X, et al. (2013b) Motion primitives and 3-D path planning for fast flight through a forest. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, pp. 2940–2947.

Paranjape AA, Meier KC, Chung SJ and Hutchinson S (2013c) Optimum spatially constrained turns for agile micro aerial

vehicles. In: *AIAA guidance, navigation and control conference*, Boston, MA, USA.

Paranjape AA, Sinha NK and Ananthkrishnan N (2008) Use of bifurcation and continuation methods for aircraft trim and stability analysis - a state-of-the-art. *Journal of Aerospace Sciences and Technologies* 60(2): 1–12.

Park S, Deyst J and How J (2007) Performance and lyapunov stability of a nonlinear path-following guidance method. *Journal of Guidance, Control and Dynamics* 30(6): 1718–1728.

Phillips WF (2009) *Mechanics of Flight*. New York: John Wiley & Sons.

Schouwenaars T, Mettler B, Feron E and How J (2004) Hybrid model for trajectory planning of agile autonomous vehicles. *Journal of Aerospace Computing, Information, and Communication* 1: 629–651.

Schouwenaars T, Mettler B, Feron E and How JP (2003) Robust motion planning using a maneuver automaton with built-in uncertainties. In: *American Control Conference*, Denver, CO.

Yang J, Dani A, Chung SJ and Hutchinson S (2013) Inertial-aided vision-based localization and mapping in a riverine environment with reflection measurements. In: *AIAA guidance, navigation, and control conference*, Boston, MA, USA.

Appendix A: Index to Multimedia Extensions

Archives of IJRR multimedia extensions published prior to 2014 can be found at http://www.ijrr.org, after 2014 all videos are available on the IJRR YouTube channel at http://www.youtube.com/user/ijrrmultimedia

Table of Multimedia Extensions

Extension	Media type	Description
1	Video	Demonstration of steady and aggressive turn primitives

Appendix B: Equations of motion

The complete state space of the motion of a rigid airplane consists of 12 variables which can be grouped into four groups.

1. Position coordinates: *x, y, z*.
2. Velocity: V, α, β, denoting the flight speed, angle of attack, and angle of sideslip.
3. Euler angles ϕ, θ_p, ψ (roll, pitch, yaw, respectively).
4. Body axis angular rates: *p, q, r*.

The primary aerodynamic coefficients are called the coefficients of lift (C_L), drag (C_D), side force (C_Y), rolling, pitching and yawing moments (C_l, C_m, C_n). The equations of motion of a rigid aircraft are given by [2008, 2011]:

$$\dot{V} = \frac{1}{m}\left(T - \frac{1}{2}\rho V^2 S C_D - mg\sin\gamma\right)$$

$$\dot{\alpha} = q - \tan\beta(p\cos\alpha + r\sin\alpha) - \frac{1}{mV\cos\beta}\left(\frac{T\sin\alpha}{V} + \frac{1}{2}\rho V^2 S C_L - mg\cos\mu\cos\gamma\right)$$

$$\dot{\beta} = p\sin\alpha - r\cos\alpha + \frac{1}{mV}\left(mg\sin\mu\cos\gamma + \frac{1}{2}\rho V^2 S C_Y - T\cos\alpha\sin\beta\right)$$

$$\dot{p} = \frac{I_y - I_z}{I_x}qr + \frac{1}{I_x}\left(\frac{1}{2}\rho V^2 S b C_l\right)$$

$$\dot{q} = \frac{I_z - I_x}{I_y}rp + \frac{1}{I_y}\left(\frac{1}{2}\rho V^2 S c C_m\right)$$

$$\dot{r} = \frac{I_x - I_y}{I_z}pq + \frac{1}{I_z}\left(\frac{1}{2}\rho V^2 S b C_n\right)$$

$$\dot{\phi} = p + \tan\theta_p(q\sin\phi + r\cos\phi)$$

$$\dot{\theta}_p = q\cos\phi - r\sin\phi$$

$$\dot{\psi} = \sec\theta_p(r\cos\phi + q\sin\phi)$$

$$\dot{x} = V\cos\gamma\cos\chi, \quad \dot{y} = V\cos\gamma\sin\chi, \quad \dot{h} = V\sin\gamma,$$

where T denotes the thrust, and the angles γ, χ and μ are obtained from

$$\begin{aligned}
\sin\gamma &= \cos\alpha\cos\beta\sin\theta_p - \sin\beta\sin\phi\cos\theta_p \\
&\quad - \sin\alpha\cos\beta\cos\theta_p\cos\phi \\
\sin\chi\cos\gamma &= \cos\alpha\cos\beta\cos\theta_p\sin\psi \\
&\quad + \sin\beta(\sin\phi\sin\theta_p\sin\psi + \cos\phi\cos\psi) \\
&\quad + \sin\alpha\cos\beta(\cos\phi\sin\theta_p\sin\psi - \sin\phi\cos\psi) \\
\sin\mu\cos\gamma &= \cos\alpha\sin\beta\sin\theta_p + \cos\beta\sin\phi\cos\theta_p \\
&\quad - \sin\alpha\sin\beta\cos\phi\cos\theta_p \\
\cos\mu\cos\gamma &= \sin\alpha\sin\theta_p + \cos\alpha\cos\theta_p\cos\phi.
\end{aligned}$$

The wing geometry is specified in terms of the span (b), chord length (c), and area (S).

The non-dimensional sideforce coefficient C_Y contains an affine contribution from the vertical tail (or rudder, as the case may be) deflection δ_r, so that

$$C_Y \equiv C_Y(\alpha, \beta, p, r) + C_{Y,\delta_r}\delta_r$$

The term $C_{Y_{\delta_r}}$ is the partial derivative of C_Y with respect to δ_r and is referred to as the "δ_r-derivative of C_Y". A similar nomenclature applies to other partial derivatives. The non-dimensional rolling and yawing moment coefficients, C_l and C_n, are given by

$$C_l = h_t C_Y, \quad C_n = -l_t C_Y$$

where $h_t > 0$ is the distance of the aerodynamic center of the vertical tail from the plane containing the wings and fuselage, while $l_t > 0$ is the distance between the center of

gravity of the aircraft and the aerodynamic center of the vertical tail. If we let $C_{Y,\delta_r} < 0$ (without loss of generality), it follows that

$$C_{n,\delta_r} > 0 \quad \text{and} \quad C_{l,\delta_r} < 0$$

We define the δ_r-derivatives of the rolling and yawing moments as

$$L_{\delta_r} = \frac{1}{2}\rho V^2 SbC_{l,\delta_r}, \quad N_{\delta_r} = \frac{1}{2}\rho V^2 SbC_{n,\delta_r}$$

where S denotes the wing area and b denotes the wing span.

Finally, we relate the body axis yaw rate r to the steady turn rate $\dot{\chi}$, assuming $\dot{\alpha} = \dot{\beta} = \dot{\mu} = \dot{\gamma} = \beta = 0$. A complete discussion is beyond the scope of this paper, but it can be shown that

$$\begin{bmatrix} p \\ q \\ r \end{bmatrix} = \begin{bmatrix} \cos\alpha & 0 & -\sin\alpha \\ 0 & 1 & 0 \\ \sin\alpha & 0 & \cos\alpha \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\mu & \sin\mu \\ 0 & -\sin\mu & \cos\mu \end{bmatrix} \begin{bmatrix} \cos\gamma & 0 & -\sin\gamma \\ 0 & 1 & 0 \\ \sin\gamma & 0 & \cos\gamma \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \dot{\chi} \end{bmatrix}$$
$$= \begin{bmatrix} -(\cos\alpha\sin\gamma + \sin\alpha\cos\mu\cos\gamma)\dot{\chi} \\ \sin\mu\cos\gamma\,\dot{\chi} \\ (\cos\alpha\cos\mu\cos\gamma - \sin\alpha\sin\gamma)\dot{\chi} \end{bmatrix}$$

Appendix C: Time-delay approximation of first-order systems

Consider a system given by

$$\dot{x} = ku, \dot{u} = a(u_c - u) \tag{31}$$

where $a > 0$ and k are constants. The above system represents an integral dynamics coupled with a first-order actuator, similar to the turn dynamics in (2) and (3). The control input u_c is modelled as a unit step function. We want to replace the actuator dynamics with a delayed step input to the integrator system.

The response of the system with a step input u_c is given in the Laplace domain by given by

$$x(s) = \frac{ka}{s^2(s+a)}, \quad x(0) = u(0) = 0$$

so that

$$x(s) = \frac{k}{s^2} - \frac{k}{as} + \frac{k}{a(s+a)} \Rightarrow x(t) = kt - \frac{k}{a}(1 - e^{-at})$$

If we replace the actuator dynamics with a time-delayed step input $u(t \geq \tau) = 1$ and $u(t < \tau) = 0$, we get $x(t \geq \tau) = k(t - \tau)$ and $x(t < \tau) = 0$. We would like to choose the time delay τ to minimize the error between the value of $x(t)$ obtained using the time-delayed input and that obtained with the actuator dynamics. It is easy to see that $\tau = 1/a$ is necessary to achieve a zero steady state error; if the error need to be zero for a specific value of t, say t_f, then we can choose $\tau = \frac{1-e^{-at_f}}{a}$. In particular, as a becomes large (i.e. as the actuator dynamics become much faster than the duration of the maneuver), it suffices to choose the limiting value $\tau = 1/a$.