Robust Hybrid Control for Autonomous Vehicle Motion Planning

Emilio Frazzoli * Munther A. Dahleh † Eric Feron ‡

Abstract

The operation of an autonomous vehicle in an unknown, dynamic environment is a very complex problem, especially when the vehicle is required to use its full maneuvering capabilities, and to react in real time to changes in the operational environment. A possible approach to reduce the computational complexity of the motion planning problem for a nonlinear, high dimensional system, is based on a quantization of the system dynamics, leading to a control architecture based on a hybrid automaton, the states of which represent feasible trajectory primitives for the vehicle. This paper focuses on the feasibility of this approach: the structure of a Robust Hybrid Automaton is defined and its properties are analyzed. Algorithms are presented for time-optimal motion planning in a free workspace, and in the presence of fixed or moving obstacles. A case study involving a small autonomous helicopter is presented: a nonlinear control law for maneuver execution is provided, and a robust hybrid automaton is constructed. Simulations showing the effectiveness of the approach are presented and discussed.

1 Introduction

In the past few years considerable interest has been shown and relevant resources have been devoted, from industry, government and academia, to the design, development and operation of autonomous aerial, underwater, and ground vehicles. The possibility of removing human pilots from danger, and the size and cost advantages of autonomous vehicles are indeed very attractive, but very often have to be compared with the performance that can be attained by human-piloted vehicles, in terms of mission capabilities, efficiency, and flexibility.

The operation of an autonomous vehicle in an unknown, dynamic and potentially hostile environment is a very complex problem, especially when the autonomous vehicle is required to use its full maneuvering capabilities, and to react in real time to changes in the operational environment. A common way of dealing with highly complex systems is via a hierarchical decomposition of the activities to be performed by the autonomous vehicles, and consequently the introduction of a hierarchy of control and decision layers (see for example [1, 2], and references therein).

The dynamics of an autonomous vehicle are inherently continuous, and as such are described by ordinary differential equations. However, digital computers are typically used for control purposes, which

*Research Assistant, Laboratory for Information and Decision Systems, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, email:frazzoli@mit.edu

†Professor, Laboratory for Information and Decision Systems, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, email: dahleh@lids.mit.edu

‡Associate Professor, Laboratory for Information and Decision Systems, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, email: feron@mit.edu

means that the overall control system will be discrete in nature. The control layers that are closest to the plant are characterized by time quantization (discrete-time control systems). The bandwidth of these control systems is usually very high, and in some cases they are seen, for design purposes, as sharing the continuous nature of the underlying plant. Higher control layers are most often designed as logical decision-making agents, operating on a discrete state space. Systems that include both discrete and continuous dynamics are usually referred to in the literature as hybrid systems. Hybrid control systems have been the object of a very intense and productive research effort in the recent years, which has resulted in the definition of very general frameworks (for example, see [3, 4] and references therein). General hybrid systems, derived from arbitrary hierarchical decompositions, however, can be extremely hard to analyze and verify, and only limited results can be obtained [5, 6]. On the other hand, it may be convenient to design the hybrid system in such a way that it offers safety and performance guarantees by construction, at least in an idealized situation. This reflects the designer's insight into the nominal behavior of the system [7, 8, 9]. Consequently, the analysis and verification problems of the hybrid system are translated to a robustness analysis problem, which can be solved using the relevant tools from systems and control theory [10, 11, 12].

In addition to the already mentioned references, other hybrid control architectures have been developed in the recent past for control of aerospace vehicles, in particular helicopters, tilt-rotor aircraft, and Vertical/Short Take-Off and Landing (V/STOL) vehicles. These vehicles are characterized by completely different dynamics during hovering and forward flight, making the introduction of a hybrid control structure a natural choice. The applications of hybrid control to aerospace vehicles found in the literature can be roughly divided into two main classes. A first class includes those applications in which the hybrid system plays the role of a flight mode manager: a discrete logic governs transitions between behavioral modes like take-off, hover, cruise, and search [13, 8, 14]. The formal verification of this kind of hybrid systems can be extremely hard, and is often foregone in favor of simulations [15]. Other hybrid system architectures proposed for air traffic conflict resolution have a rigorous mathematical foundation, but do not provide feasible trajectories for the system [11, 16], and as such cannot be applied for motion planning at scales at which the dynamics of the vehicle are important. In a second class of applications the main objective of the hybrid control architecture is ensuring the correct execution of maneuvers involving transitions between different operating conditions, by controller switching. In these papers the main objective is the tracking of a pre-planned trajectory [17, 18], maintaining the vehicle inside an assigned flight envelope [19, 20].

In the present work, the hybrid controller is responsible for both the generation and the execution of a feasible trajectory, or flight plan, satisfying the mission requirements while optimizing some performance criterion. Our main objective is the definition of a robust control architecture, and algorithms, to address the motion planning problem for an autonomous vehicle, in the presence of obstacles, and exploiting to the maximum extent the vehicle's dynamics. Even though our main focus is control of autonomous vehicles, the concepts that we will introduce can be used profitably for control of a large class of nonlinear systems: the presentation of the control architecture will be kept at a general level. In section 2 the main component of our controller, that is a robust hybrid automaton, is introduced. In section 3, control policies on the hybrid automaton are developed, both for an empty workspace and in the presence of obstacles. In section 4, a case study concentrating on motion planning for a small autonomous helicopter will demonstrate the general concepts on this particularly challenging application. A new trajectory tracking control law for underactuated systems, capable of aggressive manuevers, will be introduced. Optimal control in a free environment, and motion planning in the presence of moving obstacles will be addressed and simulation results will be provided.

2 Robust Hybrid Automaton

A possible approach to reduce the computational complexity of the motion planning problem for a nonlinear, high dimensional system, is based on a quantization of the system dynamics, in the sense that we restrict the feasible nominal system trajectories to the family of time-parametrized curves that can be obtained by the interconnection of appropriately defined primitives. These primitives will then constitute a "maneuver library" from which the nominal trajectory will be constructed. Instead of solving an optimal control problem over a high-dimensional, continuous space, we will solve a mixed integer programming problem, over a much smaller space. This paper will focus on studying the practical feasibility of this approach.

At the core of the control architecture lies a hybrid automaton, the states of which represent feasible trajectory primitives for the vehicle. Each constituent subsystem of the hybrid controller will be the agent responsible for the maneuver execution. The task of the automaton will be the generation of complete, feasible and "optimal" trajectories, via the interconnection of the available primitives. Apart from the reduction in computational complexity, one of the objectives of this approach is the ability to provide a mathematical foundation for generating a provably stable hierarchical system, and for developing the tools to analyze robustness in the presence of uncertainty in the process as well as in the environment.

We want to characterize trajectory primitives in order to: (i) capture the relevant characteristics of the vehicle dynamics; (ii) allow for the creation of complex behaviors from the interconnection of primitives (we want to obtain "good" approximations to optimal solutions) (iii) determine the *minimal* set of key parameters identifying the state of the system: this is even more important for extension to multi-vehicle operations, or more complex systems. In the following we will define the class of systems we want to control, and, accordingly, we will give a characterization of the trajectory primitives we will consider. This will be used to present the Robust Hybrid Automaton structure.

2.1 System dynamics

We will consider a time-invariant nonlinear system the dynamics of which are described by the differential equation:

$$\frac{dx}{dt} = f(x, u, w) \tag{1}$$

where $x \in X$ is the state, belonging to an n-dimensional manifold X, and u and w represent respectively the control and disturbance input signals, taking values in the sets $U \subseteq \mathbb{R}^m$, $W \subseteq \mathbb{R}^l$; both signals are assumed to be bounded with respect to some norm p, i.e. $u(\cdot) \in \mathcal{U} \subseteq \mathcal{L}_p^m$, $w(\cdot) \in \mathcal{W} \subseteq \mathcal{L}_p^l$. Finally, the function $f : X \times U \times W \to TX$ is assumed to be locally Lipschitz in its arguments. Let us concentrate first on the nominal system:

$$\dot{x} = f_0(x, u) = f(x, u, 0) \tag{2}$$

that is the system obtained from (1) when the disturbance input is identically zero. We can define equilibrium points for the nominal system (2) as the points $(\bar{x}, \bar{u})$ for which $f_0(\bar{x}, \bar{u}) = 0$. In general, we will be more interested in systems that have symmetries, for which we can define the notion of relative equilibrium.

2.2 Symmetries and relative equilibria

Roughly speaking, a symmetry on the system (2) is a group action on the state that leaves the dynamics invariant. A simple definition that is enough for the purpose of this paper is given in the following: for further details and a precise definition in a Hamiltonian mechanics framework, see [21, 22, 23, 24].

Assume that the state space has the structure of a Lie group, and that the manifold X can be written *at least locally* as $X = Y \times Z$, so that the state of the system can be written as $x = (y, z) \in Y \times Z$. Consider the map $\Psi_h : X \to X$, parametrized by an element $h \in Y$, such that the state $x = (y, z)$ is transformed into $\Psi_h(y, z) = (yh, z)$. Given the initial conditions x_0 at $t = 0$, let the time-parametrized curve $\phi_u(\cdot, x_0)$ be the resulting trajectory of the nominal system (2) when the control input is any given signal u.

Definition 1 (Symmetry group for a controlled dynamical system) *If for all initial conditions $x_0 \in X$, $y \in Y$, $t \in \mathbb{R}$ the following holds:*

$$\Psi_y \circ \phi_u(t, x_0) = \phi_u(t, \Psi_y(x_0)) \tag{3}$$

then Y is a **symmetry group** *for the system (2).*

If we indicate with $\mathfrak{y}$ the Lie algebra associated with Y we can define:

Definition 2 (Relative equilibria for controlled dynamical systems) *Assume that Y is a symmetry group for the system (2), and that it is possible to find constants $\bar{z} \in Z$, $\bar{u} \in U$ and $\hat{\eta} \in \mathfrak{y}$ such that:*

$$\phi_{\bar{u}}\left(t, (e_Y, \bar{z})\right) = (\exp(\hat{\eta}t), \bar{z}) \tag{4}$$

The resulting class of trajectories of the system $\phi_{\bar{u}}(t, (y_0, \bar{z}))$ is called a **relative equilibrium**, *or* **trim trajectory**.

The collection of all possible trim trajectories defines a manifold $\mathcal{S} \subset X \times U$, denoted as the **trim surface**. It is also clear that relative equilibria include trivially all the equilibrium points of the system. A consequence of the symmetry of the dynamics is that we can treat all trajectory primitives as equivalence classes, and choose a prototype for each primitive, starting at a reference position on the symmetry group. Without loss of generality, we can define all trajectory primitives as starting at the identity element e_Y of the symmetry group Y.

Example: A very simple example of a system with symmetries is a system with integrators:

$$\begin{array}{rcl} \dot{y} & = & z \\ \dot{z} & = & f_z(z, u) \end{array} \tag{5}$$

where $(y, z) \in \mathbb{R}^n$, and the group operation we are interested in is the usual vector addition. It is evident that a translation $\psi_{\Delta y} : (y, z) \mapsto (y + \Delta y, z)$ does not change the dynamics of the system. Relative equilibria are all trajectories for which we can find $\bar{z}$ and $\bar{u}$ such that $\dot{z} = f_z(\bar{z}, \bar{u}) = 0$. The momentum can be identified in the constant vector $\bar{z}$, and the dynamics of y on a trim trajectory are given by $y(t) = y(t_0) + \bar{z}(t - t_0)$.

A more interesting kind of symmetry, that is invariance to translation and rotation about a vertical axis, is exhibited by a large class of mechanical systems. This is true of most human built vehicles: vehicles

are designed such that once we learn how to operate them at some location (e.g. at the driving school grounds), we can apply the same skills to drive anywhere in the world. In the following we will focus on the definition of a control architecture for autonomous vehicles, however the concepts and methods are valid and can be used for systems with multiple equilibria, and possibly with relative equilibria.

2.3 Autonomous vehicle dynamics

The dynamics of a large class of small autonomous vehicles can be adequately described by the rigid body equations [25]. The configuration of the vehicle will be described by an element g of the Special Euclidean group in the three-dimensional space, usually denoted by $SE(3)$. Using homogeneous coordinates, a matrix representation of $g \in SE(3)$ is the following:

$$g = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix} \tag{6}$$

where $R \in SO(3)$ is a rotation matrix and $p \in \mathbb{R}^3$ is a translation vector. The kinematics of the rigid body are determined by

$$\dot{g} = g\hat{\xi} \tag{7}$$

where $\hat{\xi}$, denoted as *twist*, is an element of the Lie algebra $\mathfrak{se}(3)$ associated with $SE(3)$. A matrix representation of an element $\hat{\xi} \in \mathfrak{se}(3)$ is

$$\hat{\xi} = \begin{bmatrix} \hat{\omega} & v \\ 0 & 0 \end{bmatrix} \tag{8}$$

where ω and v are respectively the angular and translational velocities in body axes, and the skew matrix $\hat{\omega}$ is the unique matrix such that $\hat{\omega}u = \omega \times u$, for all $u \in \mathbb{R}^3$ The full state of the vehicle as a rigid body will then be represented by $x = (g, \hat{\xi})$, with $X = SE(3) \times \mathfrak{se}(3)$. The dynamics equations, in matrix notation, will be given by:

$$J_b\dot{\omega} = -\omega \times J_b\omega + M_b(g, \hat{\xi}, u, w) \tag{9}$$

$$m\dot{v} = -\omega \times mv + F_b(g, \hat{\xi}, u, w) \tag{10}$$

where J_b and m are the vehicle's inertia tensor and mass, and M_b and F_b represent the torques and forces in body axes, which are in general a function of the vehicle state $x = (g, \hat{\xi})$, of the control inputs u, and of the disturbances w. Note that in the above we have no assumption on the characteristics of the forces acting on the vehicle (i.e. we do not require potential forces).

The dynamics of a vehicle, including cars, aircraft, ships, etc., under fairly reasonable assumptions, (such as homogeneous and isotropic atmosphere, and constant gravity acceleration for an aircraft) are invariant to translation and rotation about a vertical axis, i.e. an axis parallel to the local gravitational acceleration. If this is the case, the subgroup $H \subset SE(3)$, composed of translation and rotations about the vertical axis is a symmetry group for the vehicle dynamics. An element $h \in H = \mathbb{R}^3 \times S^1$ is completely described by the translation vector $p \in \mathbb{R}^3$ and the heading angle $\psi \in [0; 2\pi)$.

2.4 Equilibrium points and trim trajectories

The simplest possible motion primitive is trivially represented by equilibrium points. In a system with multiple equilibrium points each equilibrium point can be chosen as a trajectory primitive. A closely

related and more interesting class of primitives is given by trim trajectories. In an autonomous vehicle setting, these can be seen as those trajectories along which the velocities in body axes (the twist) and the control input are constant.

From the above discussion of the symmetry properties, all trim trajectories will be the composition of a constant rotation $\bar{g}$ and a screw motion $h(t) \in H$, given by the exponential of an element $\hat{\eta}$ of the Lie sub-algebra $\mathfrak{h} \subset \mathfrak{se}(3)$. This screw motion corresponds in the physical space to a helix traversed at a constant speed and sideslip angle. For aerial vehicles, such helices are usually described by the parameter vector $\bar{T} := [V, \gamma, \dot{\psi}, \beta]$, where V is the magnitude of the velocity vector, γ is the flight path angle, $\dot{\psi}$ is the turning rate and finally β is the sideslip angle [26].

To make the above clearer, we will give some details on the matrix representation. The elements of H can be represented in matrix notation as:

$$h = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & x \\ \sin\psi & \cos\psi & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

The identity element e_H will be represented by the identity matrix I_4. Trim trajectories are described by the corresponding element $\hat{\eta}$ in the Lie algebra $\mathfrak{h}$, which can be represented as:

$$\hat{\eta} = \begin{bmatrix} 0 & -\dot{\psi} & 0 & V\cos\beta\cos\gamma \\ \dot{\psi} & 0 & 0 & V\sin\beta\cos\gamma \\ 0 & 0 & 0 & V\sin\gamma \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{12}$$

Following a trim trajectory for a time interval Δt results in a displacement:

$$h_{trim}(\Delta t) = \exp(\hat{\eta}\Delta t) = \begin{bmatrix} \cos\Delta\psi & -\sin\Delta\psi & 0 & r(\sin\Delta\psi\cos\beta + \cos\Delta\psi\sin\beta - \sin\beta) \\ \sin\Delta\psi & \cos\Delta\psi & 0 & r(\sin\Delta\psi\sin\beta - \cos\Delta\psi\cos\beta + \cos\beta) \\ 0 & 0 & 1 & V\sin\gamma\Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

Where $\Delta\psi := \dot{\psi}\Delta t$ and we have introduced the (signed) turning radius $r := V\cos\gamma/\dot{\psi}$.

If we restrict the motion of the vehicle to a horizontal plane (as in the case of a ground vehicle, a surface vessel, or an aircraft at constant altitude), the class of trim trajectories becomes simply the set of circle arcs on the plane, including infinite radius turns (that is straight lines). Note also that while for many kinds of vehicles, such as car-like robots and fixed wing aircraft, β is usually assumed to be zero, or very small (no skidding, or coordinated flight), this is not necessarily true for vehicles like helicopters, especially for low velocity regimes (e.g. helicopters can move sideways and backwards). It should be mentioned that usually for a given choice of $\hat{\eta}$, several choices of $\bar{g}$ and $\bar{u}$ are possible, and the selection of desirable values for them is the outcome of some off-line design process. The first step in the design of our control architecture is the selection of a number of trim trajectories. The selection of trim trajectories can be carried out by gridding the set of attainable values of $\bar{T}$; this set is compact, and can be identified with the flight envelope in the case of aerial vehicles.

This class of trajectory primitives has been used widely to construct switching control systems, in which point stabilization is achieved by switching through a sequence of controllers progressively taking the system closer to the desired equilibrium [27, 28, 29, 30]. The ideas of gain scheduling and of Linear Parameter Varying (LPV) system control can also be brought into this class [31], as well as other integrated guidance and control systems for UAV applications [32]. However, such a design choice generally results

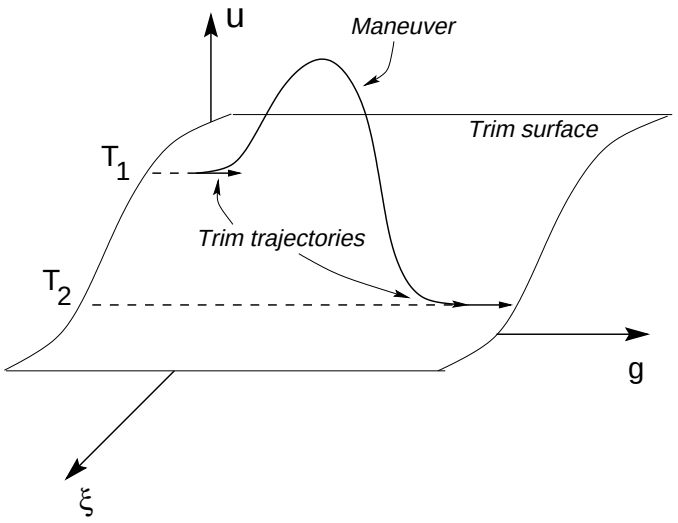


Figure 1: Trajectory primitives

in relatively poor performance, and in "slow" transitions, as the system is required to stay in some sense close to the trim surface. Moreover, the absence of any information on the transient behavior can lead to undesirable effects, such as limit cycles.

2.5 Maneuvers

For more aggressive maneuvering it is deemed necessary to better characterize trajectories that move "far" from the trim surface. Even though ours can be seen as a reductive definition of what is considered a maneuver in the common language, it leads to significant simplifications in the design of the control architecture.

Definition 3 (Maneuver) *In this paper, a* **maneuver** *is defined as a (finite time) transition between two trim trajectories, for the nominal system (2).*

Note that the transition can also be from and to the same trajectory (e.g. in the case of aircraft acrobatic maneuvers like loops and barrel rolls can be considered as transitions from and back to straight and level flight, and in the case of cars a lane change is a transition from and back to forward motion) The execution of the maneuver results in a total configuration change g_m, that is $g(t_m) = g_m g(0)$ if t_m indicates the maneuver duration. For reasons that will be made clear in the following, we are more interested in the evolution on the subgroup H, and from the properties of trim trajectories we have that $g_m = (h_m, \bar{g}_{end}\ \bar{g}_{start}^{-1})$. We will not discuss the details of how to generate the nominal state and control trajectories describing maneuvers: several methods can be used depending on the application at hand, the desired performance, and the available computing, simulation and experimental resources. Among these methods we can mention actual tests or simulations with human pilots [33], off-line solutions to optimal control problems, or real-time trajectory generation. A problem in the off-line generation of trajectories is the large amount of storage memory required; a possible solution is represented by some form of compression of the trajectory data. In this case, we have to identify some relevant parameters, on the basis of which the on-board processor can compute "easily", in real-time, a reference trajectory to track. A very efficient representation of trajectories can be achieved by exploiting the properties of differentially flat systems [34, 35].

Finally, we would like to remark that the design of the nominal trajectories, along with the tracking control introduced in the next section, has to be carried out in such a way to ensure that the vehicle does not violate constraints on its dynamic envelope (e.g. maximum velocity) [19]. Thus in this sense the objective of envelope protection is ensured implicitly by the maneuver definition.

2.6 Tracking control and disturbance rejection

In the preceding sections we defined the trajectory primitives as feasible, possibly but not necessarily optimal (for some cost) trajectories for the *nominal* system, that is when the disturbance signal w is identically zero. In real applications, we will not be able to achieve exactly the reference trajectories defined by the primitive library, because of deviations in the initial conditions, noise in the measurements, unmodeled dynamics and modeling errors, and exogenous inputs. We will therefore need to examine the behavior of the system at non-nominal conditions, and make sure that the resulting system trajectories are in some sense "close" to the trajectories of the nominal system. In general, this requires some form of feedback control, complementing the feed-forward open-loop reference input trajectory stored along with the reference state trajectory.

The reference trajectory will be completely determined, in terms of nominal state and control histories, by the primitive being executed, its inception time, and initial H configuration. A feedback control policy will then be a function $\upsilon : \mathbb{R} \times X \times H \to U$, designed to track (or regulate to) the nominal trajectory. Once a feedback control law is associated with the system (1) it is transformed into the closed-loop form:

$$\dot{x} = f(x, \upsilon(t - t_0, x, h_0), w) \tag{14}$$

in which the only exogenous input is the disturbance input w.

The robustness characteristics of equilibrium points and of trim trajectories can be expressed in terms of invariant sets.

Definition 4 (Invariant set) *A set $M \subseteq X$ is said to be a (right)-invariant set if for all $x_0 \in M$, $w \in \mathcal{W}$, and $t > t_0$:*

$$\Psi_{\exp(-\hat{\eta}(t-t_0))} \circ \phi_{\upsilon,w}(t - t_0, x_0) \in M \tag{15}$$

where $\phi_{\upsilon,w}(\cdot, x_0)$ describes the trajectory of the system under the action of the control policy υ and disturbance w, and with initial conditions $x(0) = x_0$.

Invariant sets are, properly speaking, "tubes" centered on trim trajectories. However, in the following we will refer to the section at $h = e_H$.

Definition 5 (Limit set) *We will call the limit set of a trim trajectory q the smallest invariant set $\bar{\Omega}_q$ associated with that trajectory.*

Definition 6 (Recoverability set) *will call the recoverability set $\bar{\mathcal{R}}_q$ of a trim trajectory the largest set for which there exists a finite time $\tilde{t}$ such that for all initial conditions $x_0 \in \bar{\mathcal{R}}_q$, and for all disturbance signals $w \in \mathcal{W}$, the system enters the limit set, that is if :*

$$\Psi_{\exp(-\hat{\eta}(t-t_0))} \circ \phi_{\upsilon,w}(t, x_0, t_0) \in \Omega_q, \ \forall\, t > t_0 + \tilde{t} \tag{16}$$

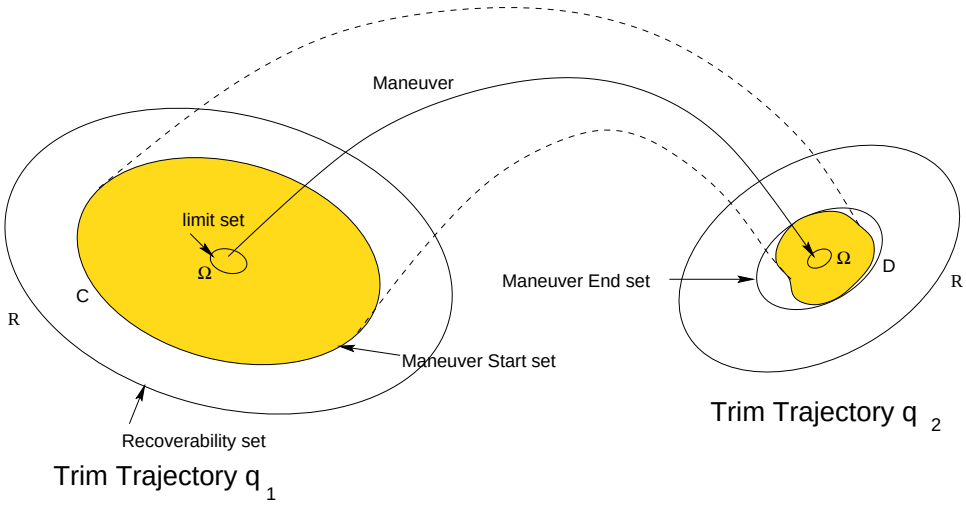


Figure 2: Invariant set definitions

In general, the exact determination of the sets $\bar{\mathcal{R}}_q$ and $\bar{\Omega}_q$ presents a very difficult challenge. However, often it is possible to compute conservative approximations, in the sense that we can compute a set $\mathcal{R}_q \subseteq \bar{\mathcal{R}}_q$ such that for all initial conditions in $\mathcal{R}_q$ the system trajectory will enter a set $\Omega_q \supseteq \bar{\Omega}_q$ after a finite time, and stay in Ω_q thereafter. It is obviously of interest to design a control law in such a way to have a large $\mathcal{R}_q$, and a small Ω_q. In the case in which the control law provides global stability, $\mathcal{R}_q$ will coincide with X, and in the case in which we have asymptotic stability, Ω_q will collapse to the trajectory itself (this is only possible if the disturbance signal w vanishes).

Similar concepts, close in nature to Lyapunov stability theory, cannot be defined for the maneuvers, since these are by definition objects with a finite time horizon. Instead we will use a concept more closely related to Poincaré maps.

Definition 7 (Image of a set) *We will define the image of a set C under the maneuver q the smallest set $D \in X$ such that for all $x_0 \in C$, and for all disturbance signals $w \in \mathcal{W}$ (supported on $[t_0, t_0 + \Delta t_{m_q}]$), we have that:*

$$\phi_{\upsilon,w}(t_0 + \Delta t_{m_q}, x_0, t_0) \in D \tag{17}$$

The objective of the control law in this case is to make the ending set D as small as possible for a given starting set C. Notice that we are not directly interested in the transient behavior of the system during the execution of the maneuver, as long as we can ensure that at the end of the maneuver the state enters the set D.

2.7 Robust Hybrid Automaton definition

We are now ready to discuss the details of the control architecture. It should be clear by now that the controlled system will include both continuous and discrete dynamics, thus belonging to the realm of hybrid control. In the following we will present the definition of a hybrid system that is based on the general model in [3]. The Robust Hybrid Automaton we are concerned with is described by the n-tuple:

$$\mathbf{RHA} = \{Q, H, T, X, f, \eta, \upsilon, \mathbf{C}, \mathbf{D}, \mathcal{R}, \Omega, \mathbf{A}, V\} \tag{18}$$

- $Q := Q_M \cup Q_T$ is the discrete set of the index state. The values of Q identify the trajectory primitive being executed; as such, Q is assumed to be a finite set. The subscript T and M indicate, respectively, trim trajectories and maneuvers;
- H is the symmetry group, identifying the position of the current trajectory primitive;
- $T = \mathbb{R}$: we augment the reference state by a clock, or timer state;
- X is the state space of the continuous system;
- f is the Lipschitz function describing the continuous system dynamics in the usual ODE form;
- $\eta := \{\eta_q, q \in Q_T\}$, where each η_q describes the motion on the trim trajectory q;
- $\upsilon := \{\upsilon_q, q \in Q\}$ is the set of control laws designed for each trajectory primitive; we assume that each υ_q is a Lipschitz function;
- $\mathbf{C} := \{C_q, q \in Q_M\}$: the collection of sets from which we can initiate maneuvers (controlled jump set);
- $\mathbf{D} := \{D_q, q \in Q_M\}$: the collection of sets at which maneuvers are terminated;
- $R := \{\mathcal{R}_q, q \in Q_T\}$: the collection of recoverability sets for each trim trajectory;
- $\Omega := \{\Omega_q, q \in Q_T\}$: the collection of limit sets for each trim trajectory;
- $\mathbf{A} = \{A_q = (q_{new}, \Delta t, \Delta h), q \in Q_M\}$: autonomous jumps occur during maneuver execution, when the timer state $\tau \in T$ reaches the value ΔT. The state is reset such that $q \leftarrow q_{new}$, $h \leftarrow h\Delta h$, $\tau \leftarrow 0$;
- $V := \mathbb{R} \times Q_M$: is the hybrid control set, determining the controlled jump execution. Controlled jumps can only be executed from trim trajectories; given a hybrid control $(\Delta t_{coast}, q_{new})$, the jump occurs when $\tau \in T$ reaches Δt_{coast}, and the state is reset such that: $q \leftarrow q_{new}$, $h \leftarrow h \exp(\eta_q \Delta t_{coast})$, $\tau \leftarrow 0$;

We can graphically depict the hybrid automaton as a directed graph, where the nodes represent the trim trajectories, and the edges represent the maneuvers. Each edge can be labeled with a cost corresponding to the maneuver duration.

The introduction of a time clock state could seem pointless, as we will in the following assume $\dot{\tau} = 1$ (when not switching). This corresponds to a classical formulation of a trajectory tracking control. The reason for introducing the clock state is that we want to allow for *maneuver regulation* [36, 32]. In that case, we will have to introduce some condition of the form $\dot{\tau}_i = \theta_i(\tau_i, x, x_{ref,i}) \geq \epsilon > 0$, to ensure that the system does not move "backwards" along a trajectory.

The RHA architecture that we have just defined can be considered as both a design paradigm and a modeling tool for nonlinear systems. The selection of the trajectory primitives, and the design of the tracking control law have to be carried out according to some specific requirements that are directly derived from the RHA structure. In particular, the conditions for the automaton consistency and controllability have to be satisfied. An example of this process will be given in the case study in section (4). On the other hand, a RHA can be seen as a powerful modeling tool, encoding all the relevant information on the dynamics of the system in a reduced set of state variables. The design of control laws for "higher-level" tasks to be performed by the system will then be substantially simplified: as will be shown in the motion planning section, it will be possible to operate in a a relatively small "maneuver-space", as opposed to the full state space. Motion planning on this maneuver space will be completely free from all the stability concerns because these have been already addressed in the construction of the RHA.

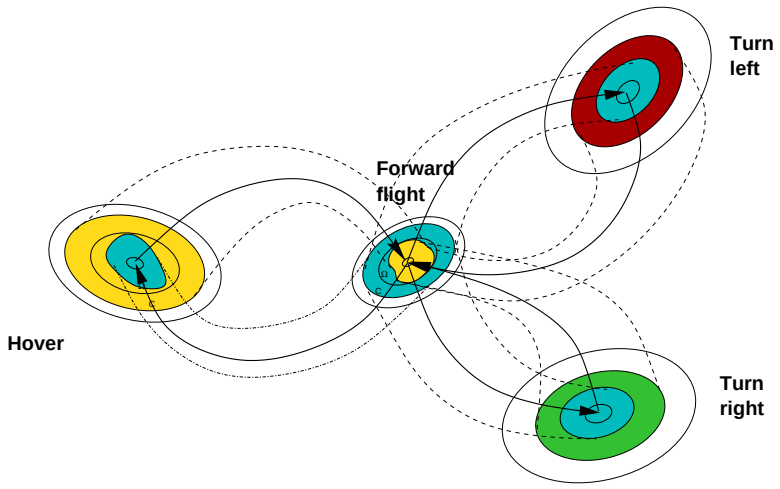


Figure 3: Robust Hybrid Automaton (simplified)

2.8 Well-posedness, consistency, and controllability

When dealing with systems of the form (1), where the right-hand side is not continuous, care must be taken to ensure that the system is well-posed, that is, a unique solution exists. In our case well-posedness is ensured by the fact that the system is piecewise continuous, and the maneuvers are a finite set of primitives with a finite time duration (by definition). As a consequence in every finite time interval there will be a finite number of switches, or discontinuities in the feedback map.

In order to decouple stability and motion planning concerns we have to ensure that the automaton is *consistent*, which means that any sequence of hybrid controls will generate a trajectory which remains "close" to the nominal trajectory, in the sense of the invariant sets defined in the previous sections.

Definition 8 (Consistency) *We say that the automaton is* consistent *if for all $q \in Q_T$ the following conditions hold:*

1. $\Omega_q \subseteq C_l,\ \forall l \in L_q$;
2. $\bigcup_{p \in P_q} D_p \subseteq \mathcal{R}_q$;
3. $\bigcup_{p \in P_q} D_p \subseteq C_l,\ \forall l \in L_q$,

where $L_q, P_q \subset Q_M$ are respectively the set of indices of maneuvers leaving and arriving at the trim trajectory q.

Remark 2.1 (Maneuver recovery) *If the first and second conditions are satisfied, the third one can always be satisfied by adequately extending the maneuvers with a recovery phase at the new trim trajectory.*

Definition 9 (Hybrid State) *We will say that the system is in the* hybrid state (q, h, τ) *if $\Psi_{h^{-1}} x \in \bigcap_{l \in L_q} C_l \subseteq R_q$, that is if the continuous state is inside the starting sets for all the maneuvers leaving the current trim trajectory.*

The full state of the system will then be described by (q, h, τ, x), where $q \in Q, h \in H, \tau \in \mathbb{R}, x \in X$.

Once we have established that the hybrid system is well posed and consistent, we have to ensure that it is controllable:

Definition 10 (Controllability) *We say that the RHA is* controllable *if it is possible to find an admissible sequence of primitives such that we can steer the system from any initial condition (q, h) to any desired location $\bar{h} \in H$, and at any desired operating condition $\bar{q} \in Q_T$, in finite time.*

It is clear that a necessary condition for controllability is that the directed graph describing the automaton be fully connected. Moreover, the set of trim trajectories must be rich enough that by interconnecting an appropriate sequence of them we cover the group H. In the case of a system with integrators, this translates simply to the requirements that the set $\{\eta_q\}, q \in Q_T$ is a complete basis for $\mathbb{R}^n$. The problem of assessing the minimum set of trim trajectories based on which we can build a controllable automaton for autonomous vehicles recalls some classical problems both in the nonlinear control and robotics literature. For example, it is known that for models of car-like robots optimal trajectories are indeed composed by straight lines and by arcs of minimum radius circles (trim trajectories in the plane)[37, 38, 39]. We are not aware of results in the literature which are applicable to systems switching between trim trajectories in a three-dimensional space, with arbitrary transients (maneuvers) during the switches. However, we can state the following:

Proposition 2.1 (Minimum set of trim trajectories) *Assume that the system dynamics are invariant to translation and rotations about a vertical axis, i.e. that H is a symmetry group for the system. Then two trim trajectories q_1 and q_2, described by the parameters $\left\{V_1, \dot{\psi}_1, \gamma_1, \beta_1\right\}$, $\left\{V_2, \dot{\psi}_2, \gamma_2, \beta_2\right\}$, along with any two maneuvers connecting them, are sufficient for controllability in a three-dimensional space if: $V_1\dot{\psi}_2 \cos\gamma_1 \neq V_2\dot{\psi}_1 \cos\gamma_2$, and $V_1 \sin\gamma_1 < 0 < V_2 \sin\gamma_2$.*

Proof: Given in the appendix. ∎

Remark 2.2 (Controllability in the horizontal plane) *If the motion of the vehicle is restricted to a horizontal plane ($\gamma_1 = \gamma_2 = 0$), then the vehicle is controllable if just the first condition above is satisfied.*

It is clear that this would be just a minimum set of trim trajectories to ensure controllability: for practical applications, the set of trim trajectories will be much richer. Notice that the "uncontrollable" car in [40], which can only turn left with different turning radii, is in fact controllable according to our definition, even though it is not small-time controllable [41].

3 Motion planning

The hybrid control architecture lends itself to computationally efficient solutions of many problems of interest for practical applications. Most often the results that are achievable using nonlinear control theory involve stability or tracking performance. Ensuring the feasibility of solutions with respect to non-convex constraints such as obstacles in the physical workspace is in many cases of interest impossible in this framework alone. At the same time when we are dealing with systems whose dynamics is important, we cannot just deal with a simplified kinematic model, or with a discretized, logic-based controller, as the resulting trajectory will in general be unfeasible for the system.

The price that we have to pay in using the hybrid automaton is the sub-optimality of the computed solutions, owing to the fact that the stored trajectory primitives do not represent the whole dynamics of the system. However, the number of trajectory primitives stored in the automaton can be increased, depending on the available memory and computational resources, so the sub-optimality gap can be reduced to a point were it is not noticeable for practical purposes. Moreover, very often a sub-optimal solution which is computable on-line can be worth more than an optimal solution that requires computational resources only available for off-line planning.

3.1 Optimal control in a free environment

At this point the design of the hybrid controller consists of the definition of a policy μ for selecting optimal jump times and destination (maneuver) from trim trajectories. We recall that all the relevant information while in a trim trajectory are defined by the hybrid automaton state and the "position" $h \in H$ at the current trajectory inception time ($\tau = 0$). On each trim trajectory $q \in Q_T$, the discrete control set can be identified with the subset $V_q \subseteq Q_M$ containing the indices of all the maneuvers that start at q. Moreover, the timing of the jump must be decided by the hybrid controller. The policy μ will then be a mapping $\mu : Q_T \times H \to \mathbb{R}_+ \times Q_M$. Assume we want to control the system to the state $(\bar{q}, \bar{h})$, and define a running cost function $\gamma : Q \times H \times \mathbb{R}_+ \to \mathbb{R}_+$, with $\gamma(\bar{q}, \bar{h}, \cdot) = 0$. Given a policy μ we can define a total cost function:

$$J_\mu(q_0, h_0) := \int_{t_0}^{\infty} \gamma\left(q(t), h(t), \tau(t)\right) dt \tag{19}$$

where the system dynamics are governed by the continuous evolution and the jump rules given in the previous sections.

A policy μ is said to be *proper* if the above integral is finite for all initial conditions. Also, in the above we assumed that both autonomous and controlled jumps occur instantaneously, and have no cost penalty. The assumption that maneuvers are strictly finite time transitions between trim points (i.e. $\min\{(\Delta t_{man})_q\}_{q \in Q_M} > 0$) ensures that there are finite switches in finite time, and that the resulting system trajectories are well defined. If the system is controllable, then a proper policy exists.

We are interested in computing the optimal policy μ^*, that is the policy that minimizes the total cost for all initial conditions. The ensuing optimal cost will be indicated by J^*. Following dynamic programming theory [42], it can be shown that the optimal cost satisfies Bellman's equation:

$$J^*(q, h) = \min_{(\tau', q')} \left[, _T(q, h, \tau') + , _M(q', h') + J^*(q'', h'') \right] \tag{20}$$

where τ' is the delay before the commanded transition to $q' \in Q_M$, h' represents the position and heading at the start of the maneuver, and q'' and h'' represent the new state at the inception of the new trim trajectory. In the above equation, $, _T$ and $, _M$ indicate the cost associated with the trim and maneuver portions of the commanded transition. Moreover, the optimal control $(\tau', q')^*$ is the minimizer of Eq.(20). Nominal stability of the control algorithm is a consequence of the optimality condition; from the consistency conditions on the automaton we also have that the system will be driven to the limit set $\Omega_{\bar{q}}$.

We notice that the optimization requires the solution of a mixed-integer program, with one continuous variable (τ'), and one discrete variable (q'). In general, the optimal cost function is not known. However, if an initial proper policy can be devised, approximate dynamic programming algorithms, such as value or policy iteration, can be used to improve on the initial policy, and possibly get to the optimal control strategy. Moreover, since the dimension of the state space has been reduced to one discrete variable and four continuous ones, neuro-dynamic programming approximation techniques for a compact representation

of the cost function can be effectively used, making the control algorithms suitable for real-time applications [43].

3.2 Motion planning in the presence of obstacles

A very important problem, especially when controlling autonomous vehicles, is represented by motion planning in the presence of fixed or moving obstacles. We are using the expression "motion planning" as opposed to the traditional "path planning" because we want to emphasize the role of the dynamics of the system, or of non-holonomicity constraints, on the allowable feasible trajectories; this problem is also known in the literature as kinodynamic planning[44].

One of the fundamental conceptual steps in addressing a problem is the selection of the appropriate representation of the system state, constraints and of the decision variables. In the path planning literature, the most commonly encountered object underlying the algorithm design is the configuration space, that is the set of all possible collision-free configurations for the robot [45, 46]. In [44], it is suggested that a more appropriate representation for a large class of motion planning problem is via the state space, since this encodes the additional information related to the system dynamics. However, the state space of non-trivial systems is typically very large, and the "curse of dimensionality" makes the solution of motion planning problems in such large-dimension spaces computationally intractable.

The alternative that we propose, through the introduction of our hybrid control architecture, can be seen as a maneuver automaton space, in which we discretize the system dynamics, not the state space. Using this representation, the hybrid automaton already encodes all the relevant information about the system dynamics and dynamic constraints (such as non-holonomicity). The dynamic constraints must then be complemented by the configuration constraints. We can define a set $\mathcal{Q} \subseteq Q \times H$ of primitives and starting configurations, such that the resulting trajectory is collision free. The set $\mathcal{Q}$ is the direct counterpart of the traditional configuration space $\mathcal{C}$, but in addition completely encodes the system dynamic constraints.

To check whether a given trajectory is collision-free, we have to check that the appropriate invariant sets do not intersect with the obstacles. In the case of trim trajectories, the relevant set is the controlled jump set $\bigcup_{l \in L_q} C_l$. In the case of maneuvers, this would be the set spanned by the image of C_q over the maneuver duration. A perhaps more interesting and useful definition for the set $\mathcal{Q}$ is as a *safe* set of maneuvers; in the case of moving obstacles the set $\mathcal{Q}$ will be time dependent. In general, the computation of such a set is very challenging. In the hybrid system literature an approach that has been applied to several problems, including air traffic control, is based on the definition and the computation of the level sets of an appropriately defined Hamiltonian function [47, 11]. A possibly conservative computation of the safe set can however be carried out quite easily in several cases; for example, in the case of fixed obstacles, each trajectory, the execution of which allows for a complete stop along a collision free trajectory, is safe.

Notice that the set $\mathcal{Q}$ is considerably smaller than the complete state space X; moreover, the hybrid control space V is typically smaller that the continuous control space U. This means that in general, solution to optimal control problems for the hybrid automaton will be computationally less expensive than solutions on the full state space. We found that the hybrid automaton model, for its structure and properties, lends itself in a very straightforward manner to the implementation of a new class of motion planning algorithms, based on a randomized approach [48, 49, 50]. In particular, we will use a form of the so-called Rapidly-exploring Random Trees (RRT's), introduced in [51, 40].

Consider the case in which we want to control our system to the state $(\bar{q}, \bar{h}) \in \mathcal{Q}$, starting from the state $(q_0, h_0) \in \mathcal{Q}$. A concise statement of our version of the RRT algorithm is in the following:

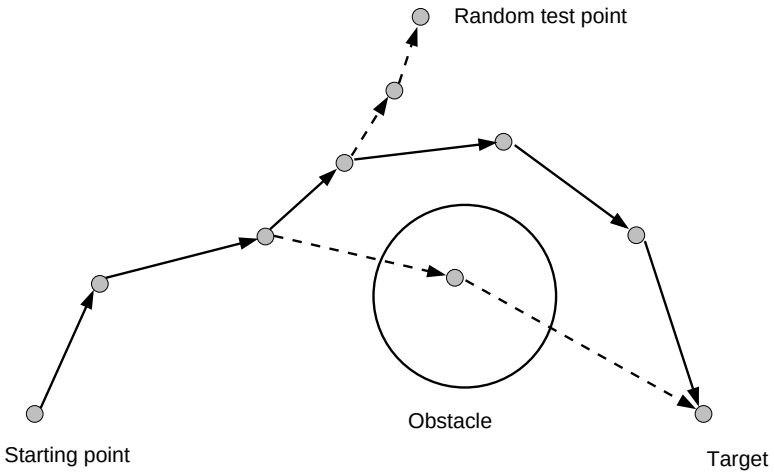


Figure 4: RRT algorithm - Pictorial representation of one iteration

1. We start building a tree, and assign (q_0, h_0) as its root. Also, set the index i to 0.

2. As the next step, we compute the optimal path in the obstacle free case: this will generate a sequence of states $\{(q_{i,1}, h_{i,1}), \dots, (q_{i,n}, h_{i,n})\}$, where $q_{i,n} = \bar{q}$ and $h_{i,n}$ can be made arbitrarily close to $\bar{h}$;

3. define $\hat{n}$ as the index of the last one of the above states that belongs to the set $\mathcal{Q}$; if $\hat{n} = n$ then we have found a feasible solution to the problem. Moreover, if $i = 0$, the solution is the optimal one (subject to the dynamics represented in the trajectory primitives library), and we can exit. Otherwise add all the states $\{(q_{i,1}, h_{i,1}), \dots, (q_{i,\hat{n}}, h_{i,\hat{n}})\}$ as a child to the node i;

4. choose (at random) a new "destination state" $(\tilde{q}, \tilde{h}) \in \mathcal{Q}$, and compute the corresponding optimal obstacle-free path, starting from the "closest" state in the tree (indexed as j). In this case, we can use as a distance function the unconstrained optimal cost function $J^*(q, h \cdot \bar{h}^{-1})$. If the first state in the computed path is in $\mathcal{Q}$, then add it to the tree as a child of the state j, and set i to the index of the newly added tree node

5. if the maximum iteration number has not been exceeded, set $i \leftarrow i + 1$ and go back to step 2, otherwise exit with a failure.

The hybrid automaton architecture solves many of the issues that could be a problem for the effectiveness of the RRT algorithm as presented in [51]. First, the optimal cost function J^* represents a valid distance function, with a meaning that is directly related to the system (e.g. time to target in a minimum time problem). Moreover, once the optimal cost function is computed, the selection of the optimal controller and hence of the optimal path can be carried out in an extremely efficient and computationally inexpensive fashion. The need for a second tree, growing from the target point, is removed by the fact that at every step the tree grows in the direction of $(\bar{q}, \bar{h})$. For an online implementation, the tree growth is not parametrized with constant time steps that have to be kept "small", but instead with the coasting times (in trim trajectories) and the maneuver durations, that are not necessarily "small". This means that the computation of the new hybrid control input value does not need to be performed at a high rate.

The RRT algorithm is not complete, in the sense that it could fail to generate a feasible trajectory, even if one exists. However, it can be shown that the RRT algorithm, in the continuous case, provides a feasible solution with high probability, and is complete in a probabilistic sense [52]. Randomized algorithms have been found to perform very well in many applications of interest; however, RRTs cannot deal very well with workspace configurations that present "narrow passages" [53]. In many cases of interest though, like in the case of aerial vehicles, the environment is such that the work space does not have such narrow passages;

moreover, in cases where this narrow passages exist, they could be known a priori, and appropriate waypoints could be set.

4 Case study: control of an autonomous helicopter

A small helicopter is a very good example of the systems to which our control architecture can be profitably applied. Radio-controlled helicopters are capable of remarkably agile and aggressive maneuvers, which are impossible to perform using traditional control techniques, especially when the on-line solution of the motion planning problem is required. Among the reasons for that we can state that helicopters are essentially unstable systems, with a very high bandwidth, and their dynamics change considerably throughout the flight envelope. We anticipated that an approach based on a quantization of the system dynamics, through the definition of a library of maneuvers and other trajectory primitives could deal effectively with the complex dynamics of a helicopter, and at the same time provide a valuable modeling tool for motion planning.

In the process of building a robust hybrid automaton for the helicopter, the problem of designing feedback control laws for stabilization and tracking had to be faced. The RHA definition does not specify any particular form for the low-level control law, so it is possible to choose the most convenient design methodology. Several techniques for autonomous helicopter control have been proposed, ranging from robust linear control, neural control, fuzzy control, and nonlinear control [54, 55, 56]. We will focus on the latter techniques, mainly because of the fact that, if we can find control Lyapunov functions for the system, we can derive very easily estimates of the invariant sets R_q and Ω_q, as well as estimates of the image of a starting set C_q under a maneuver.

In recent papers, feedback linearization techniques have been applied to helicopter models, with positive results. The main difficulty in the application of such techniques is the fact that, for any meaningful selection of outputs, the helicopter dynamics are non-minimum phase, and hence are not directly input-output linearizable. However, it is possible to find good approximations to the helicopter dynamics such that the approximate system is input-output linearizable, and bounded tracking can be achieved [57, 35, 58] The above mentioned approach suffers from a few drawbacks, including the fact that, since attitude is often parameterized using Euler angles, singularities arise when performing some maneuvers, such as loops, barrel rolls and split-S's [59]. A possible solution to the singularity problem is represented by chart switching when approaching a singularity. However, this can be cumbersome in implementation, and can lead to excessively high gains in the proximity of singularities. On the other hand, the singularities arising in these model are artifacts due to the choice of the attitude parameterization (Euler angles), and do not reflect any intrinsic characteristic of the helicopter dynamics. The need to avoid artificial singularities due the attitude representation is the main driver behind the control design presented in this paper. This can not be achieved using a coordinate-dependent control system: to achieve this goal, we will operate directly in the configuration manifold of the helicopter.

The "tracking on manifolds" problem is solved in [60] for fully actuated mechanical systems. In the following we present an extension, for achieving asymptotic (locally exponential) tracking of trajectories for a particular class of underactuated mechanical systems. An approximation of the helicopter model can be shown to be in this class: the approximation that will be shown in the paper is the same one that leads to feedback linearizability, or differential flatness of the model. However, the method presented here can deal, through trivial modifications, with more accurate models, including for example the effects of aerodynamic forces.

The control design will be based on a non-trivial extension of backstepping ideas [61, 62, 63] to dynamic

systems on manifolds. In its basic form the backstepping procedure is carried out on a chain of integrator (integrator backstepping); in our case the backstepping procedure is implemented on a dynamic system evolving on the group of rotations in the three-dimensional space $SO(3)$. A backstepping approach for control of underactuated, non-minimum phase nonlinear systems was used in [64, 65] for control of surface vessels: in our case the problem is more difficult since we need to control the rigid body motion in the three-dimensional space, as opposed to the plane. At the time of writing this paper, the authors also became aware of the work in [66], where a backstepping procedure was implemented to design a controller for a helicopter close to hover. However, the approach in the above paper is still partially based on a coordinate representation, and Euler angles are used in the expression of the control law. As a consequence, geometric singularities are not eliminated, and the system is not able to track trajectories in which the helicopter "turns upside down"[67]. In addition to providing a more rigorous approach to the "backstepping on manifolds" design procedure, our formulation avoids the introduction of artificial singularities, and results in a controller that is capable of tracking any feasible trajectory (within the limitations of the model).

4.1 Helicopter dynamics

The helicopter model that we will use here is based on the model presented in [35]. A very similar model has been widely used in the nonlinear control literature, in the three degrees of freedom case, as a VTOL aircraft model [57, 58]. More details on helicopter dynamics can be found in [68, 69, 70]. The dynamics of the helicopter as a rigid body can be expressed as in section (2.3), with the following expressions for the body force and moment in eq. (9):

$$\begin{cases} F_b &= mR^{-1}\tilde{g} + u_4 \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} + \begin{bmatrix} 0 & -\epsilon_2 & 0 \\ \epsilon_1 & 0 & -\epsilon_3 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 - \epsilon_4(u_4) \end{bmatrix} \\ M_b &= \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} \end{cases} \tag{21}$$

In the above, $\tilde{g}$ is the gravity acceleration, and we have defined:

$$\begin{cases} u_1 &:= -z_{mr} T b_1 \\ u_2 &:= -z_{mr} T a_1 \\ u_3 &:= k_0 + k_T T^{1.5} + x_{tr} F_t \\ u_4 &:= T \end{cases} \tag{22}$$

and:

$$\begin{cases} \epsilon_1 &:= -\dfrac{1}{z_{mr}} \\ \epsilon_2 &:= -\dfrac{1}{z_{mr}} \\ \epsilon_3 &:= -\dfrac{1}{x_{tr}} \\ \epsilon_4(T) &:= k_0 + k_T T^{1.5} \end{cases} \tag{23}$$

where T is the main rotor thrust, a_1, b_1 are respectively the pitch and roll rotor flapping angles, F_t is the tail rotor thrust, $-z_{mr}$ and $-x_{tr}$ are the moment arms of the main and tail rotor with respect to the helicopter center of mass, and k_0, k_T are the coefficient in the approximate expression of the main rotor reaction torque. As suggested and motivated in [35], we will use a dynamic extension procedure to ensure that the augmented system has constant relative degree. This is done by appending two integrators to the thrust control input u_4. We will thus consider as the control inputs the vector $\mathbf{u} := [u_1, u_2, u_3]^T$ and the scalar $\ddot{u}_4$.

4.2 Control Design

The objective of the feedback control law will be to track a smooth, feasible reference trajectory $(g_{ref}(t), \hat{\xi}_{ref}(t))$. We will start by designing a controller for the approximate system obtained by setting $\epsilon_i = 0, i = 1 \dots 4$. This approximation simplifies the control design considerably: it can be shown that the resulting approximate system is differentially flat, and hence feedback linearizable [35]. Even though we will not use a feedback linearization technique, the absence of unstable zero dynamics gives an insight on the nature and the advantages of the approximate system.

4.2.1 Translational dynamics

If we consider as translational coordinates the position and velocity in the inertial frame $(p, \dot{p}) = (p, Rv)$, the translational dynamics is composed of three double integrators in parallel, driven by the force input $\alpha(R, u) := RF_b(u)$. Since the translational dynamics block is essentially linear, it is easy to design a Lyapunov function V_p and a control policy $\mathcal{K}_p$ such that if

$$\alpha(R, u_4) = \bar{\alpha}(p, \dot{p}) := \mathcal{K}_p(p, \dot{p}) - m\tilde{g} \tag{24}$$

then the translational dynamics is stable, that is $\dot{V}_p(p, \dot{p}) \leq -W(p, \dot{p})$, where W is a positive definite function. The above can be easily extended for tracking of a reference trajectory $p_{ref}(t)$, by adding the appropriate feed-forward terms. For simplicity, we will assume the following proportional-derivative (PD) form for the translational dynamics control law:

$$\mathcal{K}_p(p, \dot{p}) := \ddot{p}_{ref} - K_p(p - p_{ref}) - K_{\dot{p}}(\dot{p} - \dot{p}_{ref}) \tag{25}$$

where the constant gains $K_p, K_{\dot{p}}$ can be derived from standard linear control design techniques, and $\ddot{p}_{ref}$ is the feed-forward term.

If the function $\alpha(R, u_4)$ were invertible, then we would be able to use the attitude R as a control input to the translational block. This is not the case, however, we can select the desired attitude R_d as the "closest" (in the sense explained below) element to reference attitude R_{ref} for which we can find a u_4 such that $\alpha(R_d, u_4) = \bar{\alpha}(p, \dot{p})$. A measure of the distance between two elements R_1, R_2 of $SO(3)$ can be derived from the relative rotation $\delta R := R_1 R_2^{-1}$ (group error), that is still an element of $SO(3)$ [60]. All the elements of $SO(3)$ can be described by a fixed axis $\tilde{r}$, corresponding to the single real eigenvector, and an angle of rotation $\tilde{\theta}$, which can be derived from the complex conjugate eigenvalues. As a measure of the magnitude of the group error δR, that is the distance between the rotations R_1 and R_2, we can consider the following function:

$$\Theta(\delta R) := 1 - \cos(\tilde{\theta}) = 2\sin^2\frac{\tilde{\theta}}{2} = \frac{1}{2}\mathrm{Tr}(I - \delta R) \tag{26}$$

At this point, we can define the desired attitude and thrust as the solution of the following optimization problem:

$$\begin{aligned} (R_d, u_{4d}) \quad = \quad & \underset{(R,u)\in SO(3)\times\mathbb{R}}{\arg\min} \quad \Theta(RR_{ref}^{-1}) \\ & s.t. \quad \alpha(R, u) = \bar{\alpha}(p, \dot{p}) \end{aligned} \tag{27}$$

It can be verified (see the following) that a unique solution exists, and that the dependence of (R_d, u_{4d}) on $(p, \dot{p})$ is smooth, excluding the sets over which $R_{ref}e_3 \cdot \bar{\alpha}(p, \dot{p}) = 0$. This includes the case in which the commanded acceleration of the helicopter is equal to the gravity acceleration. This singularity is inherent to the physics of the problem, and as such cannot be avoided: it corresponds to the fact that if $u_4 = 0$ the

helicopters enters a free fall, regardless of the attitude. Moreover, in the case in which the commanded acceleration requires a rotation of $\pi/2$ radians of amplitude, there are two equivalent solutions to the problem (27), corresponding to $u_{4d} = \pm\bar{u}$. Having defined the vector $r := R_{ref}e_3 \times \bar{\alpha}(p,\dot{p})/||\bar{\alpha}(p,\dot{p})||_2$, simple geometric reasoning provides the following solution to the above minimization problem:

$$(R_d, u_{4d}) = \begin{cases} \left(\mathrm{Rot}(-r, \sin^{-1}||r||_2)R_{ref},\ ||\bar{\alpha}(p,\dot{p})||_2\right), & \text{if } R_{ref}e_3 \cdot \bar{\alpha}(p,\dot{p}) \leq 0 \\ \left(\mathrm{Rot}(r, \sin^{-1}||r||_2)R_{ref},\ -||\bar{\alpha}(p,\dot{p})||_2\right), & \text{if } R_{ref}e_3 \cdot \bar{\alpha}(p,\dot{p}) > 0 \end{cases} \tag{28}$$

where $e_3 := [0,0,1]^T$, $\mathrm{Rot}(r,\theta)$ is the rotation about the fixed axis r through an angle θ, and $\cdot$ and $\times$ represent the scalar and cross products of vectors in $\mathbb{R}^3$. The rotation $\mathrm{Rot}(r,\theta)$ is given by Rodrigues' formula:

$$\mathrm{Rot}(r,\theta) = I_3 + \sin\theta\frac{\hat{r}}{||r||_2} + (1-\cos\theta)\frac{\hat{r}^2}{||r||_2^2} \tag{29}$$

4.2.2 Attitude dynamics and backstepping control design

Once we have the desired attitude, using backstepping ideas [61, 62, 63], we want to track R_d in such a way to stabilize the overall system. However, before we can go on with the control design, we have to take a look at the rate of change of the objects introduced in the previous section (see also the derivations in [60]). First of all, we have that $\dot{R}_{ref} = R_{ref}\hat{\omega}_{ref}$; accordingly, we can define $\hat{\omega}_d := R_d^T\dot{R}_d$. Furthermore, the following equalities hold:

$$\begin{aligned} \tfrac{d}{dt}\Theta(RR_d^T) &= \sin\tilde{\theta}\frac{d\tilde{\theta}}{dt} = \\ &= \mathrm{Skew}(R_d^TR)^\vee \cdot (\omega - \omega_d) = \mathrm{Skew}(RR_d^T)^\vee \cdot R_d(\omega - \omega_d) = \\ &= \sin\tilde{\theta}\ \tilde{r} \cdot R_d(\omega - \omega_d) = \sin\tilde{\theta}\ \tilde{r}_d \cdot (\omega - \omega_d) \end{aligned} \tag{30}$$

where $\mathrm{Skew}(M) = \frac{1}{2}(M - M^T)$, the operator $(\cdot)^\vee$ is the inverse of the "hat" operator (i.e. $S^\vee \times u = Su$, $\forall u \in \mathbb{R}^3$, and S is a skew 3×3 matrix), and $\tilde{r}, \tilde{\theta}$ are respectively the fixed axis and rotation angle of the attitude error RR_d^T, obtained by:

$$\cos\tilde{\theta} = \frac{\mathrm{Tr}(RR_d^T) - 1}{2} \tag{31}$$

$$\sin\tilde{\theta}\tilde{r} = \mathrm{Skew}(RR_d^T)^\vee \tag{32}$$

We can rewrite eq.(30) as:

$$\frac{d}{dt}\Theta(RR_d^T) = \nabla\Theta \cdot (\omega - \omega_d) \tag{33}$$

having defined:

$$\nabla\Theta := \sin(\tilde{\theta})\tilde{r}_d = \mathrm{Skew}(R_d^TR)^\vee \tag{34}$$

Now we are ready to state the following result (for the definition of asymptotic tracking on manifolds, see [60]):

Theorem 1 (Asymptotic tracking for the approximate system) *Given a smooth, feasible state trajectory $x_{ref}(t) = (g_{ref}(t), \xi_{ref}(t))$ for a rigid body under the action of the forces in eq. (21), with $\epsilon_i = 0, i = 1\ldots 4$, there exists an (almost everywhere) smooth control law under which the system state $x(t)$ globally asymptotically, and locally exponentially tracks the reference trajectory $x_{ref}(t)$.*

Proof: We will prove the above statement by actually building a tracking control law. Define a candidate Lyapunov function by adding to V_p terms that opportunely penalize the attitude configuration and velocity errors. Such a candidate Lyapunov function is the following:

$$V = V_p(p, \dot{p}) + k_\theta \Theta(R R_d^T) + \frac{1}{2}\left(||\eta||_2^2 + k_u||u_4 - u_{4d}||_2^2 + ||\zeta||_2^2\right) \tag{35}$$

where:

$$\eta := \omega - \omega_d - \frac{u_{4d}}{k_\theta \cos\frac{\tilde{\theta}}{2}} R_d^T \left(\tilde{t} \times \nabla_{\dot{p}} V_p\right) \tag{36}$$

$$\zeta := \dot{u}_4 - \dot{u}_{4d} - \frac{1}{k_u} \nabla_{\dot{p}} V_p \cdot R e_3 \tag{37}$$

and:

$$\tilde{t} := \frac{(R + R_d)e_3}{||(R + R_d)e_3||_2} \tag{38}$$

Computing the time derivative of V, with the definition of R_d and u_4 given in the previous section, we get:

$$\frac{dV}{dt} \leq -W_p(p, \dot{p}) + \nabla_{\dot{p}} V_p(p, \dot{p})(u_{4d} R_d e_3 - u_4 R e_3) + k_\theta \nabla\Theta \cdot (\omega - \omega_d) + \eta\dot{\eta} + k_u(u_4 - u_{4d})(\dot{u}_4 - \dot{u}_{4d}) + \zeta\dot{\zeta} \tag{39}$$

We can make the above negative semidefinite by imposing:

$$\dot{\eta} = -k_\eta \eta - k_\theta \nabla\Theta \tag{40}$$

$$\dot{\zeta} = -k_\zeta \zeta - k_u(u_4 - u_{4d}) \tag{41}$$

where k_η, k_θ, k_ζ, k_u, are all positive constants. Noting that:

$$\nabla_{\dot{p}} V_p \cdot (u_{4d} R_d e_3 - u_4 R e_3) = -\nabla_{\dot{p}} V_p \cdot [u_{4d}(R - R_d)e_3 + (u_4 - u_{4d}) R e_3] = \tag{42}$$

$$= -\nabla_{\dot{p}} V_p \cdot \left[2u_{4d} \sin\frac{\tilde{\theta}}{2}(\tilde{r} \times \tilde{t}) + (u_4 - u_{4d}) R e_3\right] = \tag{43}$$

$$= -k_\theta \nabla\Theta \cdot \left(\frac{u_{4d}}{k_\theta \cos\frac{\tilde{\theta}}{2}} R_d^T (\tilde{t} \times \nabla_{\dot{p}} V_p)\right) - k_u(u_4 - u_{4d}) \nabla_{\dot{p}} V_p \cdot R e_3 \tag{44}$$

$$= k_\theta \nabla\Theta \cdot (\eta - \omega + \omega_d) + k_u(u_4 - u_{4d})(\zeta - \dot{u}_4 + \dot{u}_{4d}) \tag{45}$$

we get:

$$\frac{dV}{dt} \leq -W_p(p, \dot{p}) - k_\eta ||\eta||_2^2 - k_\zeta ||\zeta||_2^2 \leq 0 \tag{46}$$

The time derivative along system trajectory of the Lyapunov function V is hence negative-semidefinite: asymptotic stability can be inferred from LaSalle's principle. To prove local exponential stability, augment the Lyapunov function (35) with a cross term:

$$V_{cross} = \chi\left[\nabla\Theta \cdot \eta + (u_4 - u_{4d})\zeta\right] \tag{47}$$

where χ is a positive constant. The time derivative of the cross term, under the control law (40) can be computed as:

$$\begin{aligned} \tfrac{d}{dt} V_{cross} &= \chi\left[\tfrac{d\nabla\Theta}{dt} \cdot \eta + \nabla\Theta \cdot \dot{\eta} + (\dot{u}_4 - \dot{u}_{4d})\zeta + (u_4 - u_{4d})\dot{\zeta}\right] = \\ &= \chi\left[-k_\theta ||\nabla\Theta||_2^2 - k_\eta \nabla\Theta\eta + \tfrac{d\nabla\Theta}{dt} \cdot \eta - k_u(u_4 - u_{4d})^2 - k_\zeta(u_4 - u_{4d})\zeta + (\dot{u}_4 - \dot{u}_{4d})\zeta\right] \end{aligned} \tag{48}$$

Moreover, it is possible to find positive constants c_1, c_2 so that we have the following bounds:

$$\frac{d\nabla\Theta}{dt} \cdot \eta \leq (1 + ||\nabla\Theta||_2)||\eta||_2^2 + c_1||\nabla\Theta||_2^2 \, ||\eta||_2 \, ||P||_2^2 \tag{49}$$

and:

$$(\dot{u}_4 - \dot{u}_{4d})\zeta \leq \zeta^2 + c_2||P||_2\zeta \tag{50}$$

where $P := [p - p_{ref}, \dot{p} - \dot{p}_{ref}]^T$. For sufficiently small χ, and error vector $\delta := [P, \nabla\Theta, \eta, u_4 - u_{4d}, \zeta]^T$, the derivative of the augmented Lyapunov function $V_{total} := V + V_{cross}$ will be negative definite, and it will be possible to find a $\lambda > 0$ such that $\dot{V}_{total} < -\lambda V_{total}$, which proves local exponential stability.

The control law will be smooth almost everywhere, that is for all conditions for which $\tilde{\theta} \neq \pi/2$. The explicit expression for the control torques $\mathbf{u} = [u_1, u_2, u_3]^T$, and the control force second derivative $\ddot{u}_4$ will then be given by:

$$\begin{cases} \mathbf{u} &= \omega \times \mathbb{J}\omega + \mathbb{J}\left[-k_\eta\eta - k_\theta\nabla\Theta + \frac{d\omega_d}{dt} + \frac{d}{dt}\left(\frac{u_{4d}}{k_\theta \cos\frac{\tilde{\theta}}{2}} R_d^T \left(\tilde{t} \times \nabla_{\dot{p}} V_p\right)\right)\right] \\ \ddot{u}_4 &= \ddot{u}_{4d} - k_\zeta\zeta - k_u(u_4 - u_{4d}) + \frac{1}{k_u}\frac{d}{dt}\left(\nabla_{\dot{p}} V_p \cdot Re_3\right) \end{cases} \tag{51}$$

■

To the authors' knowledge, the above control law is a new result, providing asymptotic tracking for a class of underactuated mechanical systems on SE(3). While based on the control design framework presented in [60], the control law we presented provides asymptotic tracking for a broader class of systems. The class of systems for which the control law is applicable comprises vehicles modeled as rigid bodies subject to one force in a body-fixed direction, and three independent torque components. The main advantage of (51) is the absence of artificial singularities, deriving from attitude parameterizations, like Euler angles.

Note that the elimination of geometric singularities has been accomplished through an over-parametrization of the outputs: we need to fully specify the reference attitude if we want to achieve asymptotic tracking. Of course the reference attitude has to satisfy the constraints represented by the system dynamics, for the full trajectory to be feasible. It is to be mentioned that we can also specify an unfeasible attitude reference trajectory: in that case we will not be able to achieve asymptotic tracking for the whole state. However, we can guarantee that the system trajectory will be such to asymptotically track the position reference, and the attitude will be the *closest* (in the sense of eq. (26)) element in $SO(3)$ to the reference attitude that ensures the feasibility of the trajectory. The only requirement needed is that this unfeasible reference attitude must differ from the actual feasible attitude by less than $\pi/2$ radians. As a final remark, note that imposing (28) is equivalent to requiring that a body-fixed vector (e.g. a camera line of sight) be pointed as close as possible to a specified direction. Yaw tracking only ensures that the *projection* on a horizontal plane of the body-fixed vector points in a specified direction.

4.2.3 Tracking for the actual model

So far, we have been able to design a controller to achieve asymptotic tracking of a reference trajectory for the approximate system. The terms neglected in the approximation appear as perturbations in the nominal model. A first question that arises is how will the controller designed for the approximate system behave for the actual system, and in the presence of bounded external forces $||F_e||_2 \leq \Delta_e$, like for example those ensuing from uncertainties in the aerodynamics, or from wind gusts. We want to analyze the robustness properties of the control law given in eq. (51), in order to construct a consistent automaton, as defined in section (2.8).

Assume that the reference trajectory $x_{ref}(t) = (g_{ref}(t), \xi_{ref}(t))$ is feasible for a rigid body under the forces in (21). Note that if we replace in the relevant equations in the previous paragraph the nominal thrust direction in body axes e_3 with the actual thrust direction as obtained from eq. (21, the control laws (51) will give exact tracking for initial conditions on the reference trajectory, and no external disturbances.

As a first step in our analysis we will consider trim trajectories. Since the unmodeled forces F_u are a function of the control, given in eq. (21), that is a smooth function of the states and the reference trajectory, we have that, in a compact set $\mathcal{R} := x \in X, V(x, x_{ref}) \leq \bar{V}$, we can characterize the effect of the neglected coupling as:

$$||F_u||_2 \leq \Delta_u + \epsilon||\delta||_2 \tag{52}$$

where δ is the state error vector $\delta = [p - p_{ref}, \dot{p} - \dot{p}_{ref}, \nabla\Theta, \eta, u_4 - u_{4d}, \zeta]^T$. If we assume that we can measure the acceleration of the vehicle (this is a reasonable assumption, since autonomous vehicles are usually equipped with accelerometers), we have the following:

Theorem 2 (Bounded tracking for trim trajectories) *Given a trim trajectory* $x_{ref}(t) = \left((\exp(\hat{\eta}_{ref}\, t), \bar{g}), \bar{\xi}_{ref}\right)$ *for a rigid body under the action of the forces in eq. (21), there exist sufficiently small* $\epsilon, \Delta = \Delta_u + \Delta_e$ *such the control law defined in section (4.2.2) is such that for all initial conditions in* $\mathcal{R}$ *the state* $x(t)$ *achieves bounded tracking of the reference trajectory* $x_{ref}(t)$.

Proof: If we compute the time derivative of the Lyapunov function V_{total} under the effect of the disturbance forces, we get that:

$$\begin{array}{rcl} \dot{V}_{total} & \leq & -\lambda V_{total} + \nabla V_{\dot{p}}(F_e + F_u) \leq \\ & \leq & -\lambda V_{total} + \beta||\delta||_2\ (\Delta + \epsilon||\delta||_2) \leq \\ & \leq & -(\lambda - \beta\epsilon\mu)V_{total} + \beta\Delta\sqrt{\mu V_{total}} \end{array} \tag{53}$$

where μ is such that $||\delta||_2^2 \leq \mu V_{total}$. Defining the set:

$$\Omega = \left\{ x \in X \mid V_{total}(x, x_{ref}) < \mu \left(\frac{\beta\Delta}{\lambda - \beta\epsilon\mu} \right)^2 \right\} \tag{54}$$

we have that, if $\epsilon < \lambda/(\beta\mu)$, $\dot{V}_{total}$ is negative semi-definite in the set $\mathcal{R} \setminus \Omega$. Note that this, to be of any significance, requires that Δ be small enough that $\Omega \subset \mathcal{R}$, that is $\mu \left(\frac{\beta\Delta}{\lambda - \beta\epsilon\mu} \right)^2 < \bar{V}$. Finally, notice that if $\Delta = 0$, the control law for the approximate system is asymptotically stabilizing for the actual system ■

It turns out that the values of ϵ for small helicopters are such that the conditions in the above theorem are satisfied (see simulation results in section 4.3. Theorem 2 can be used to characterize the recoverability and limit set estimates for each trim trajectory. The same Lyapunov function for the nonlinear system can be used to study the behavior of the system during a maneuver. In this case, define the set $\mathcal{C} := x \in X, V_{total}(x, x_{ref}) \leq \bar{V}$, and assume that we can characterize the effect of the neglected coupling as:

$$||F_u||_2 \leq \Delta_u + \epsilon||\delta||_2 \tag{55}$$

for $x \in \mathcal{C}$. Then, by the same procedure used earlier, we have that:

$$\dot{V}_{total} \leq -(\lambda - \beta\epsilon\mu)V_{total} + \beta\Delta\sqrt{\mu V_{total}} \tag{56}$$

After the maneuver duration t_m, we have that the state will be contained in the set:

$$\mathcal{D} := \left\{ x \in X | V_{total}(x, x_{ref}) < \left[\sqrt{\bar{V}} - \frac{\beta\Delta\sqrt{\mu}}{\lambda - \beta\epsilon\mu} \exp\left(-\frac{\lambda - \beta\epsilon\mu}{2} t_m \right) + \frac{\beta\Delta\sqrt{\mu}}{\lambda - \beta\epsilon\mu} \right]^2 \right\} \tag{57}$$

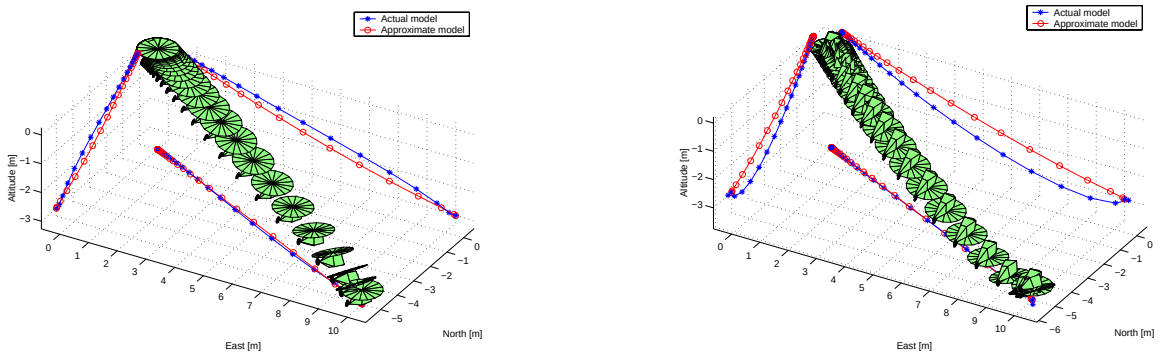


Figure 5: Point stabilization example

Notice that by extending the maneuver duration t_m we can make the measure of the set $\mathcal{D} \setminus \Omega$ arbitrarily small. The above characterization of the destination sets $\mathcal{D}$ given a starting set $\mathcal{C}$ can be used to design maneuvers. In the general case, the computation of the Poincaré map will have to be carried out by specific techniques. Computationally efficient algorithms for robustness analysis of trajectory tracking for nonlinear systems have recently been developed, and can be profitably used in this case [71, 72]. Once we have computed the Poincaré maps, we have the tools for constructing a consistent automaton as defined in Section 2.8.

As a further note, the performance bounds just derived are obtained by using a simple PD controller for the translational dynamics. However, the form of the disturbance input is such that is satisfies a matching condition: as a consequence, a robust control policy $\mathcal{K}(x)$ can be designed in a relatively straightforward manner to achieve better bounds. This possibility will be not be explored here, and is left to future research.

4.3 Simulation examples

In this section we show some simulations obtained for the tracking control law for point stabilization, trim trajectory tracking, and for tracking of "aggressive" maneuvers that cannot be handled in a straightforward manner by coordinate-based controllers because of singularities in the attitude parametrization.

The first example that we will show is a stabilization problem. The helicopter starts at hover with non-zero position coordinates, and we want to hover at the origin, heading due North. In Fig. 5 we show the response of the approximate system and of the actual system in the two cases of "normal" attitude and inverted flight. Note that in order to obtain discernible plots, we had to multiply the ϵ_i values for a typical model helicopter by a factor of 10. As we can see, the application of the controller designed for the approximate system to the actual system (with exaggerated coupling terms) gives very good results. As expected, the response of the approximate system is identical in the two cases the previous case (up to a rotation of 180 degrees). However, we see in the response of the actual system in the inverted flight case the typical "undershoot" of non-minimum phase systems. The helicopter model used in this paper is known in the literature as being non-minimum phase, where the zero dynamics can be represented as undamped oscillators, of the form $\ddot{\phi} = -k \sin \phi$. As a consequence, we notice the non-minimum phase behavior when the attitude of the helicopter is such that the zero dynamics evolve close to the unstable equilibrium point.

A third example, in fig. (6), is about tracking of a trim trajectory, in this case a climbing turn. Again we see satisfactory performance, even though we notice that tracking a time-parameterized trajectory could require excessive control effort and flying aggressiveness. Maneuver tracking techniques [36, 32] could be

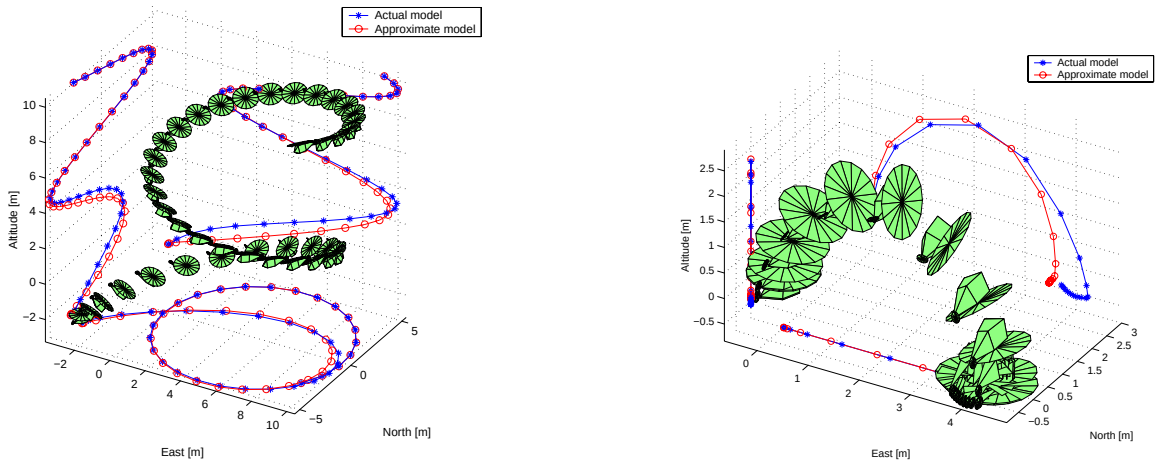


Figure 6: Trim trajectory (left) and maneuver (right) tracking example

profitably used in this case, but we will leave this to future work.

As a fourth example, always in fig. (6) we consider tracking a trajectory that performs a transition to inverted flight. This maneuver, in the case in which continuous controls are required, has to go through the singularity at $T = u_4 = 0$, since the thrust will be positive (upwards in the body frame) in the initial condition, and negative (downwards in the body frame) in the inverted flight condition. In the example, attitude control was shut off when $|u_{4_d}|$ was smaller than a preset value T_{min}. As a matter of fact, on helicopters the rotor thrust can be changed very quickly by just a step in the collective: since the kinetic energy stored in the rotor can be used to provide very fast commanded thrust responses, the thrust dynamics are much faster than the rigid body dynamics. Hence the requirement of having a continuous thrust history can be relaxed.

In the transition to inverted flight maneuver that we are considering, we are close to the singular condition only for a short period of time, during which we cannot guarantee that $\dot{V} < 0$. This is the reason for the relatively large deviations from the reference trajectory in the second half of the maneuver (following the singularity). However, the increment ΔV between before and after the singularity can be made to be arbitrarily small (for example by reducing T_{min}, at the expense of a larger required control authority).

4.4 Time optimal control in a free environment

For this application example, we consider the minimum time optimal control problem, in an obstacle-free environment. We want to take a helicopter to hover in a neighborhood of the origin in minimum time, under the constraint of the allowable maneuvers. In this case the running cost function is:

$$\gamma(q,h) = \begin{cases} 0 & \text{for } (q,h) = (q_{hover}, [\bar{x}, \cdot]^T), ||\bar{x}|| < \epsilon \\ 1 & \text{otherwise} \end{cases} \tag{58}$$

The radius of the target zone ϵ can be made arbitrarily small, but must be strictly positive, at least in the current implementation of this architecture, because of truncation (finite number of trajectory primitives), and computational issues (continuity at the optimum). As an alternative an additional terminal cost can be included in the cost expression. As a simplifying assumption, we will consider only trajectories in the horizontal plane. In this case the problem has an axial symmetry, and the relevant information in the outer state vector can be reduced to the scalar quantities ρ and λ, that is the distance and the line-of-sight angle to the target (see fig. 7). In the example, the design trim trajectories collection is defined by:

$$
\begin{aligned}
(V, \dot{\psi}, \gamma, \beta) \quad \in \quad & \{0, 1.25, 2.5, 5, 10 \text{ m/s}\} \times \\
& \times \{-1, 0.5, 0, 0.5, 1 \text{ rad/s}\} \times \\
& \times \{0 \text{ rad}\} \times \{0 \text{ rad}\}
\end{aligned}
$$

Reference maneuvers are computed for transition between all trim trajectories. Each maneuver is computed by connecting the trim parameters by splines of sufficiently high order to guarantee the required smoothness in the control inputs, and looking for the spline of least duration while satisfying constraints on the states, controls, and control rates.

An initial proper control policy, based on heuristics, can easily be derived (i.e. stop the helicopter, turn facing the target, move slowly towards the target). Application of a value iteration algorithm provides convergence in the evaluation of the optimal cost-to-go to within one hundredth of a second in 15 iterations. In this application example, the computation of the optimal cost is carried out off-line (see fig. 8). The evaluation of the optimal control, that is the computation of the minimizer (the arg min) of J as defined in eq. (20), which has to be done in real-time, requires only a few hundredths of a second on a Pentium-class CPU, and is therefore implementable on current on-board computer systems for small aerial vehicles. Examples of trajectories obtained by simulation are shown in fig. 9. In these figures, the height of the stems represents the velocity of the vehicle; moreover, solid lines and circle symbols indicate transitions in the hybrid automaton.

4.5 Motion planning with obstacles

As explained in section (3), the optimal cost function computed in the obstacle-free case can be profitably used in a randomized path-planning algorithm. A variation of the algorithm summarized in section(3) has been implemented to allow for selection of the best feasible trajectory, still considering time as the performance measure. Essentially, such a modification consists in back-propagating the time to target each time a feasible child trajectory is added to the tree. By back-propagating we mean that we have to climb the tree back towards the root, labeling each node as the time to target of the child tree plus the time required for the transition from the node to the root of the child tree. This climbing process has to be reiterated until the label on the node under examination is actually smaller that the new computed value (this means that the current trajectory is not the best one found so far), or until we get to the maneuver tree root, in which case the current trajectory represents the best choice. Also, notice that care must be taken in labeling the nodes that belong to obstacle-free solutions as such, and exclude them from the randomized search. This is because an obstacle-free solution is optimal by construction, so we should not waste computing time in trying to improve it.

The randomized path planning has been tested in several examples, including cases with moving obstacles, and proved to be very fast and reliable. In the two cases depicted in fig. (10), the randomized motion planner succeeded in finding 50-100 feasible trajectories, for a running time ranging between 2 and 6 seconds.

5 Conclusions

In this paper a Robust Hybrid Automaton architecture, applicable to autonomous vehicles has been presented and discussed. Algorithms have been given to solve, or approximate, the time-optimal motion planning problem in the free workspace case as well as in the presence of fixed and moving obstacles. A specific example, involving a small autonomous helicopter, has been presented in detail. The hybrid automaton structure has been found to provide a very flexible, and computationally effective tool for

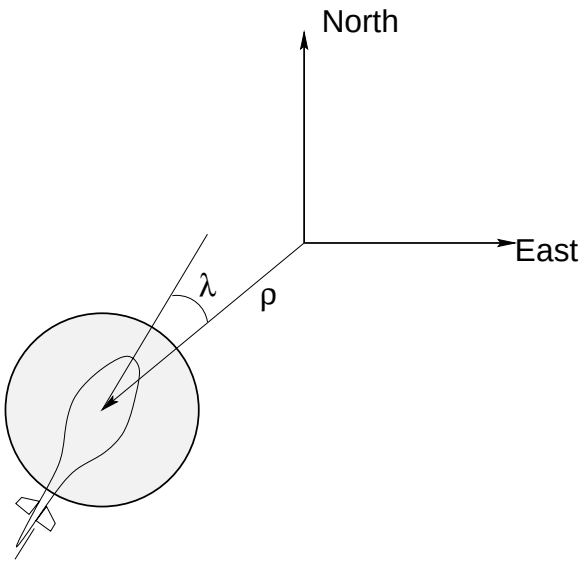


Figure 7: Example geometry

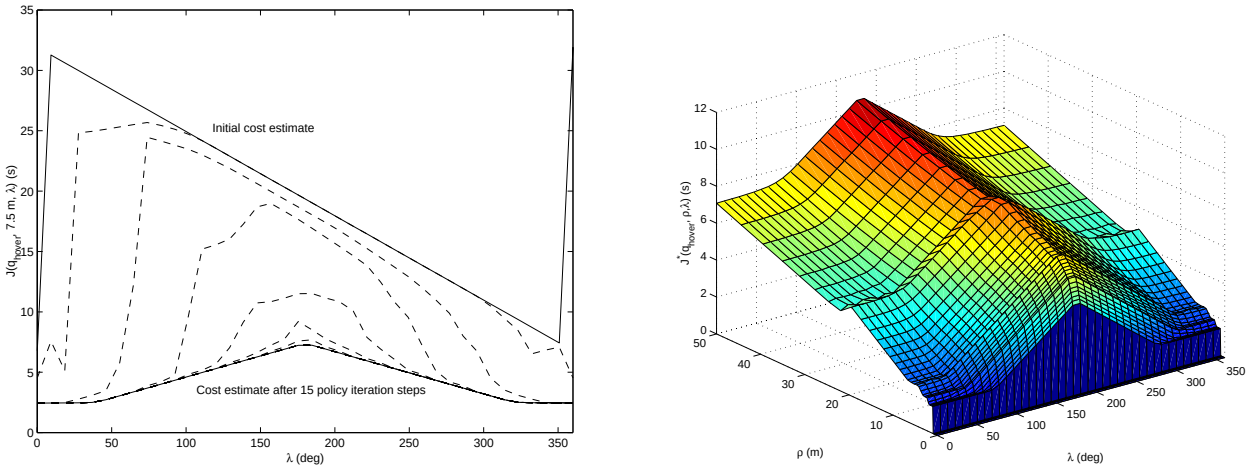


Figure 8: Value iteration results and optimal cost

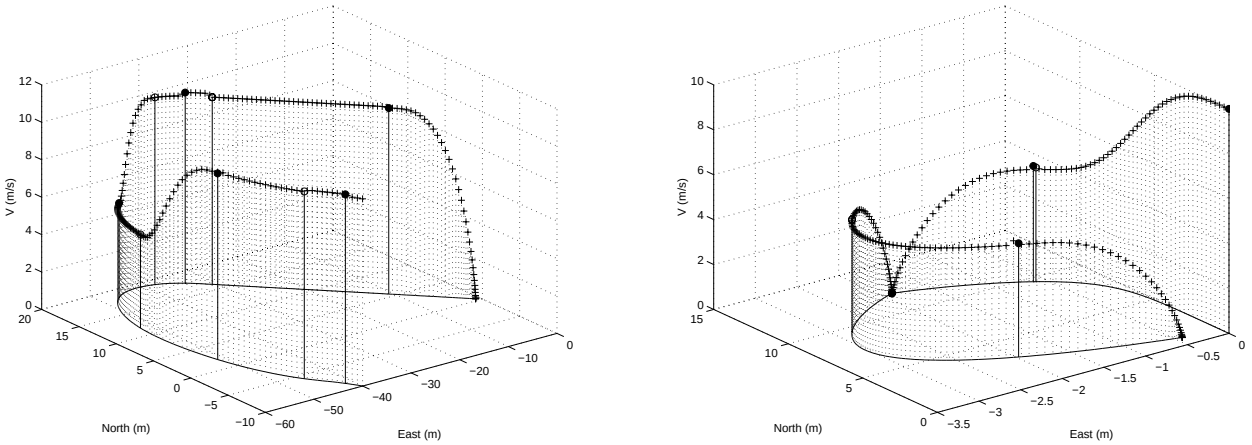
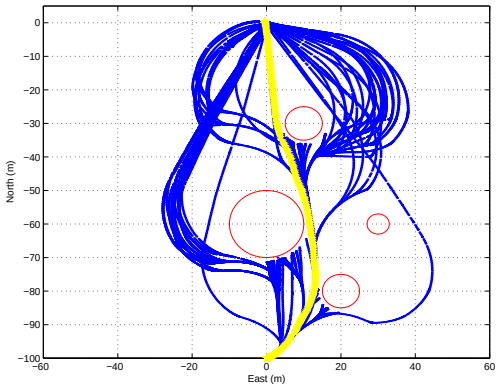


Figure 9: Simulated trajectory and velocity profile, starting from a high speed turn away from the target (left), and from high speed flight over the target


North (m)
East (m)
0
−10
−20
−30
−40
−50
−60
−70
−80
−90
−100
−60
−40
−20
0
20
40
60