Robust Hybrid Control for Autonomous Vehicle Motion Planning

by

Emilio Frazzoli

Dottore in Ingegneria Aeronautica
Università degli Studi di Roma "La Sapienza" (1994)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2001

Author .....
Department of Aeronautics and Astronautics
May 25, 2001

Certified by .....
Munther A. Dahleh
Professor of Electrical Engineering
Thesis Supervisor

Certified by .....
Eric Feron
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by .....
Brent Appleby
Principal Member of the Technical Staff, Draper Laboratory
Thesis Advisor

Certified by .....
Brian C. Williams
Associate Professor of Aeronautics and Astronautics
Thesis Advisor

Accepted by .....
Wallace E. Vander Velde
Professor of Aeronautics and Astronautics
Chairman, Department Committee on Graduate Students

Robust Hybrid Control for Autonomous Vehicle Motion Planning

by

Emilio Frazzoli

Submitted to the Department of Aeronautics and Astronautics
on May 25, 2001, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This dissertation focuses on the problem of motion planning for agile autonomous vehicles. In realistic situations, the motion planning problem must be solved in real-time, in a dynamic and uncertain environment. The fulfillment of the mission objectives might also require the exploitation of the full maneuvering capabilities of the vehicle.

The main contribution of the dissertation is the development of a new computational and modelling framework (the Maneuver Automaton), and related algorithms, for steering underactuated, nonholonomic mechanical systems. The proposed approach is based on a quantization of the system's dynamics, by which the feasible nominal system trajectories are restricted to the family of curves that can be obtained by the interconnection of suitably defined primitives. This can be seen as a formalization of the concept of "maneuver", allowing for the construction of a framework amenable to mathematical programming.

This motion planning framework is applicable to all time-invariant dynamical systems which admit dynamic symmetries and relative equilibria. No other assumptions are made on the dynamics, thus resulting in exact motion planning techniques of general applicability. Building on a relatively expensive off-line computation phase, we provide algorithms viable for real-time applications.

A fundamental advantage of this approach is the ability to provide a mathematical foundation for generating a provably stable and consistent hierarchical system, and for developing the tools to analyze the robustness of the system in the presence of uncertainty and/or disturbances.

In the second part of the dissertation, a randomized algorithm is proposed for real-time motion planning in a dynamic environment. By employing the optimal control solution in a free space developed for the maneuver automaton (or for any other general system), we present a motion planning algorithm with probabilistic convergence and performance guarantees, and hard safety guarantees, even in the face of finite computation times.

The proposed methodologies are applicable to a very large class of autonomous vehicles: throughout the dissertation, examples, simulation and experimental results are presented and discussed, involving a variety of mechanical systems, ranging from simple academic examples and laboratory setups, to detailed models of small autonomous helicopters.

Thesis Supervisor: Munther A. Dahleh
Title: Professor of Electrical Engineering

Thesis Supervisor: Eric Feron
Title: Associate Professor of Aeronautics and Astronautics

Acknowledgments

First and foremost, I would like to express my gratitude to my two co-advisors, Prof. Munther Dahleh and Prof. Eric Feron. They have been a constant source of inspiration, advice, and encouragement during all these years, both in my academic life and beyond. I benefited greatly from the unique perspective that each of them brought into the research leading to this work. In particular, I would like to thank Munther for sharing his profound insight in systems and control theory, and for his endless words of encouragement. I would like to thank Eric for his uncountable suggestions, and his openness and thoroughness in discussing and evaluating new ideas and research directions. Moreover, his enthusiasm has been a constant driving force behind my research work.

I would like to acknowledge the constant support from Dr. Brent Appleby and Dr. Marc McConley of the Draper Laboratories. They provided the motivation for most of the work in this dissertation, and were always helpful in suggesting research directions to pursue, and feedback on my progress. I am grateful to Prof. Brian Williams, Dr. James Paduano, and Prof. Charles Coleman, for their availability, and for their willingness to serve on my thesis committee and provide advice in their areas of expertise.

I would like to thank Prof. Raffaello D'Andrea of Cornell University and Prof. Nicola Elia, now at Iowa State University, for many stimulating discussions, and for providing many suggestions which led to significant advances in my research. During my years at MIT I also came in contact with a large number of other preeminent scholars and researchers, who shaped my education. I benefited particularly from interaction with Prof. John Deyst, Prof. Alexander Megretski, and Prof. Sanjoy Mitter.

It is fair to say that I have learned almost as much from other students as I have from faculty. One of the most luminous examples of scholarship I have ever met is Joseph Saleh. Every word I have exchanged with him has left me enriched in more ways than academics. I am honored to count him among my best friends. Jerry Wohletz was always a solid source of advice and insight in everything regarding aircraft dynamics and control, not to mention geopolitics. Richard Kornfeld has always been a constant and reliable source of wisdom and advice, even after his departure from MIT. I am grateful to Vladislav Gavrilets, for sharing his expertise in the design and development of unmanned air vehicles. I will never forget the discussions on nonlinear systems with Reza Olfati-Saber, until early in the morning. Other fellow students and friends whom I would like to acknowledge include Jae-Hyuk Oh, Zhi-Hong Mao, Kazutaka Takahashi, Jan De Mot, David Dugail, Lucia Pallottino, Sean Warnick, Jorge Goncalves, Soosan Beheshti, Constantinos Boussios and Georgios Kotsalis.

The administrative staff of the Aeronautics and Astronautics Department and of LIDS were always helpful and supportive, often above and beyond the call of duty. In particular, Marie Stuppard, Fifa Monserrate, and Doris Inslee made my life at MIT much easier.

I am grateful to the past and current members of the MIT European Club executive committee, for making my life at MIT much more pleasant and exciting (and busy). Antonio Bruno, Ignacio Perez de La Cruz, Cyril Morcrette, Sven Heemeyer, Gerhard (Gags) Schick, and Markus Palenberg, to name just a few, have left a permanent mark in my heart.

I owe everything I am and everything I have ever achieved to my parents, Franco Vittorio Frazzoli and Carla Perrone. I will always be grateful to them for having given me all that is important in life: from their unrelenting love and unconditional support, to the values of integrity and hard work they have taught me through their own example.

One last word of gratitude goes to Maria Romana Cassì, for her courage and for her smile. Thank you for being here for me.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In recent years, considerable interest has been shown in, and relevant resources have been devoted to, the design, development and operation of unmanned aerial, underwater, and ground vehicles. Unmanned spacecraft have been orbiting the Earth, and other objects in the solar system, for decades.

The purposes of such unmanned vehicles are extremely diverse, ranging from scientific exploration and data collection, to provision of commercial services, and military reconnaissance and intelligence gathering. Other areas of applications could in the future include law-enforcement, search and rescue, and even entertainment.

Unmanned vehicles make it possible to perform critical tasks without endangering the life of human pilots. Moreover, the designers have more freedom in the development of the vehicle, not having to account for the presence of a pilot, and the associated life-support and safety systems. This potentially results in cost and size savings, as well as increased operational capabilities, performance limits, and stealthiness.

At present, most of such systems are operated remotely by human operators, from dedicated control stations. However, as the capabilities of such systems develop, and their complexity reaches unprecedented levels, there is a strong perceived need for an increased level of automation, in order to improve the system's efficiency, reliability, and safety, and to reduce the total life cycle cost.

1.1 Motion Planning

A basic problem which has to be faced and solved by autonomous vehicles, on which this dissertation will focus, is the problem of *motion planning*. By motion planning we mean the generation and execution of a plan for moving from one location to another location in space to accomplish a desired task, while at the same time avoiding collisions with obstacles or other undesirable behaviors. Moreover, it is desirable that the plan makes optimal use of the available resources to achieve the goal optimizing some "cost" measure.

In realistic situations, the motion planning problem must be carried out in real-time, in a dynamic, uncertain and possibly hostile environment. The fulfillment of the mission objectives might also require the exploitation of the full maneuvering capabilities of the vehicle.

The problem of motion planning has attracted the interest of different communities, including researchers in the area of nonlinear control, robotics and artificial intelligence. In the last 30 years considerable effort has been devoted to research in this area, and has

resulted in a dramatic improvement of the capabilities of motion planning algorithms. The interest is motivated both by the relevance of the problem to practical applications and its theoretical challenges.

1.1.1 Problem Formulation

In this section we will define the classes of problems that we will address in the rest of the dissertation. To clearly define the problems we are interested in, we need to introduce briefly some concepts and notation which will be discussed in more detail in Chapter 2.

The motion planning problems we will consider are defined over a time-invariant mechanical control system $\mathcal{S}$ (representing for example the dynamics of a vehicle), whose state is completely determined by a set of parameters x, the *state*, belonging to a *state space* $\mathcal{X}$. The state of the vehicle evolves over time, as the vehicle moves subject to the control inputs u, according to a law which is usually expressed through a set of Ordinary Differential Equations (ODEs), such as $dx/dt = f(x,u)$. Starting from some initial condition $x(t_0) = x_0$, and under the action of a given control signal $u : t \mapsto u(t)$, the state at time t can be written as $x(t) = \phi_u(t, x_0)$. The function ϕ_u represents the *state flow* of the system; it is also known as *transition function.*

The basic problem of steering of a mechanical system can be stated as follows:

Problem 1.1 (Motion Planning in a Free Environment) *Given a mechanical control system $\mathcal{S}$, and the initial and final conditions $x_0, x_f \in \mathcal{X}$, find a control input signal $u : t \mapsto u(t)$ such that $x_f = \phi_u(t_f, x_0)$.*

In many cases of interest, though, the choice of possible input signals is limited in the sense that there are some constraints on the evolution of the state of the system, for example to ensure safe operation of the vehicle (e.g. the flight envelope constraints for aircraft and rotorcraft, speed limits for ground vehicles). Such constraints can typically be encoded by inequalities of the form

$$F\left(x(t), u(t)\right) \leq 0, \tag{1.1}$$

where F can be a vector of constraints, and the inequality should be read component-wise. This kind of constraints, to which we will refer as *flight envelope constraints*, is closely related to the vehicle and its dynamics. For this reason, we can also refer to this class constraints as "internal constraints". What this means in the context of this dissertation is that flight envelope constraints share some of the fundamental properties of the dynamical system, namely its symmetries (to be defined in section 2.2). We can now formulate a more meaningful problem:

Problem 1.2 (Constrained Motion Planning in a Free Environment) *Given a mechanical control system $\mathcal{S}$, and the initial and final conditions $x_0, x_f \in \mathcal{X}$, find a control input signal $u : t \mapsto u(t)$ such that $x_f = \phi_u(t_f, x_0)$, and $F\left(\phi_u(t, x_0), u(t)\right) \leq 0$, for all $t \in [t_0, t_f]$.*

In the above two problems, we are not concerned with the "quality" of the generated trajectory, as long as it satisfies the boundary conditions, and the flight envelope constraints. However, for practical applications, it is desirable to characterize the quality of the solution according to a meaningful performance measure, usually referred to as "cost" of the generated trajectory. The nature of the motion planning problem will thus move from that a *feasibility* problem to that of an *optimization* problem. Examples of widely used

cost functions include: the time required for the execution of the trajectory, its length, the deviation from a reference trajectory, control effort, or a combination thereof. All of these performance measures can be obtained by defining a cost functional on the state-control trajectory, as follows:

$$J(x(\cdot), u(\cdot)) := \int_{t_0}^{t_f} \gamma(x(t), u(t))\, dt, \tag{1.2}$$

and biasing the search of feasible trajectory towards cost-efficient trajectories. Hence we have the new problem:

Problem 1.3 (Optimal Motion Planning in a Free Environment) *Given a mechanical control system $\mathcal{S}$, the initial and final conditions $x_0, x_f \in \mathcal{X}$, and a cost functional $J(x(\cdot), u(\cdot))$, find a control input signal $u : t \mapsto u(t)$ such that $x_f = \phi_u(t_f, x_0)$, $J(\phi_u(\cdot, x_0), u(\cdot))$ is a global minimum for J, and $F(\phi_u(t, x_0), u(t)) \leq 0$, for all $t \in [t_0, t_f]$.*

Up to this point, we have not considered "external constraints" on the vehicle, that is, constraints that are not directly related to the vehicle itself and its dynamics, and as such do not share its properties. Such external constraints can encode obstacle avoidance requirements, where we can give the word "obstacle" a broad meaning to include physical obstacles and other vehicles, as well as threats, no-fly zones, or other regions in the state space which are "forbidden" to the vehicle. Such *obstacle avoidance* constraints can be encoded as a set of pointwise in time, possibly time-varying, constraints on the state of the vehicle, of the form

$$G(x(t), t) \leq 0. \tag{1.3}$$

Notice that the dependence on time allows the modelling of moving obstacles.

In certain applications of interest, we can also have integral constraints on the state and the input, of the form

$$\int_{t_0}^{t} I(x(\tau), u(\tau))\ d\tau \leq 0. \tag{1.4}$$

The most obvious of such constraints is given by fuel consumption. As additional examples of integral constraints we can mention: stealthiness requirements, introduced to keep the probability of detection along a trajectory within some limits, and thermal control requirements on spacecraft (e.g. to avoid pointing radiators towards the Sun for too long).

We can now formulate a very general class of motion planning problems as follows:

Problem 1.4 (Motion Planning) *Given a mechanical control system $\mathcal{S}$, and the initial and final conditions $x_0, x_f \in \mathcal{X}$, find a control input signal $u : t \mapsto u(t)$ such that $x_f = \phi_u(t_f, x_0)$, $F(\phi_u(t, x_0), u(t)) \leq 0$, $G(\phi_u(t, x_0), t) \leq 0$, and $\int_{t_0}^{t} I(\phi_u(\tau, x_0), u(\tau))\ d\tau \leq 0$, for all $t \in [t_0, t_f]$.*

The problem outlined above is just a feasibility problem, since its solution requires the computation of any trajectory satisfying the constraints and the boundary conditions. Most often, in practical applications, we are also interested in optimizing some performance measure, for example in terms of a cost functional such as the one given in (1.2):

Problem 1.5 (Optimal Motion Planning) *Given a mechanical control system $\mathcal{S}$, initial and final conditions $x_0, x_f \in \mathcal{X}$, and a cost functional $J(x(\cdot), u(\cdot))$, find a control input signal $u : t \mapsto u(t)$ such that $x_f = \phi_u(t_f, x_0)$, $J(\phi_u(\cdot, x_0), u(\cdot))$ is minimized, $F(\phi_u(t, x_0), u(t)) \leq 0$, $G(\phi_u(t, x_0), t) \leq 0$, and $\int_{t_0}^{t} I(\phi_u(\tau, x_0), u(\tau))\ d\tau \leq 0$ for all $t \in [t_0, t_f]$.*

1.1.2 Nonlinear Control

The nonlinear control community has concentrated on the problem of steering underactuated, maximally nonholonomic (i.e. controllable) mechanical systems in a free environment, that is on Problems 1.1–1.3. Application of nonlinear control tools and ideas to this set of problems generated powerful and elegant methods for steering a robot between arbitrary configurations, which will be very briefly described in the following.

Optimal Control

Perhaps the best formulated general method for addressing motion planning problems is the use of optimal control [21, 6]: through the introduction of an arbitrary cost functional Problems 1.1 and 1.2 can always be reduced to Problem 1.3. A recent survey of such an approach for trajectory generation appeared in [12]. In the following we will concentrate on a local coordinate-based representation of the systems dynamics; a coordinate-independent expositions of the optimal control problem for mechanical systems on Lie groups can be found in [63, 139].

Given a cost functional of the form (1.2), the basic steering problem, including constraints on the input, can be tackled using techniques from variational calculus, by forming a controlled Hamiltonian, in the following way (dropping the explicit dependence on time of x,u, and λ):

$$\mathcal{H}(x, \lambda, u) = \gamma(x, u) + \lambda^T f(x) \tag{1.5}$$

where the vector λ is known as the costate vector. Under suitable technical assumptions (e.g. continuous differentiability of the Hamiltonian [6, 78]), the optimal controls are determined by the application of the celebrated Minimum Principle of Pontryagin [106], according to which the optimal control minimizes the Hamiltonian $\mathcal{H}$. Denoting the optimal state and control trajectories by the superscript $(\cdot)*$, we have

$$u^* = \arg\min_{u \in \mathcal{U}} \mathcal{H}(x^*, \lambda^*, u). \tag{1.6}$$

The equation above, together with the condition $\mathcal{H}(x^*, \lambda^*, u^*) = 0$, the initial and final conditions on x, and the state and costate dynamics:

$$\begin{aligned} \frac{dx^*}{dt} &= \frac{\partial \mathcal{H}}{\partial \lambda}, \\ \frac{d\lambda^*}{dt} &= -\frac{\partial \mathcal{H}}{\partial x} \end{aligned}$$

completely determines the optimal trajectory. If an optimal control law u^* is used, the minimum value of the cost functional (1.2) is a function only of the initial state, expressing the optimal cost-to-go $J^*(x)$ from the initial conditions x. The optimal cost-to-go function is also referred to as optimal return function in the literature [21].

A related methodology is based on Bellman's principle of optimality [9], according to which any portion of an optimal trajectory is also an optimal trajectory. The principle of optimality is the foundation of dynamic programming [11], and in our case leads, under the assumption of continuous differentiability of the optimal cost-to-go function $J^*(x)$, to the following version of the Hamilton-Jacobi-Bellman equation:

$$\min_{u \in \mathcal{U}} \left(\gamma(x, u) + \nabla J^*(x)^T f(x, u) \right) = 0, \tag{1.7}$$

(where $\nabla J^*(x)$ is the gradient of $J^*(x)$), and the following expression for the optimal control:

$$u^* = \arg\min_{u \in \mathcal{U}} \left(\gamma(x,u) + \nabla J^*(x)^T f(x,u) \right). \tag{1.8}$$

Notice that from the above equation we can infer the equivalence between the costate vector λ and the gradient of the optimal cost-to-go $J^*(x)$ (and hence the sensitivity of the cost-to-go to changes in the initial conditions).

The same approaches, based on variational calculus, the Pontryagin's principle, and dynamic programming, can be used, at the cost of some additional complexity in the set up of the equations, to solve problem including state constraints.

The equations presented in this section provide, in principle, a computable solution to the full optimal control problem. However, practical implementation of these methods for non-trivial systems, and especially for real-time applications, is far from being feasible.

The variational approach requires the solution of a two-point boundary value problem, which has to be done numerically, without any guarantees on convergence and convergence rate; a very good initial guess is usually required, and still the problem can be considerably difficult from the numerical point of view. Using the collocation approach [28, 12], the two-point boundary value problem can be converted into a nonlinear program; however, the resulting nonlinear program is non-convex, and as a result, also in this case, no guarantees on the convergence and convergence rate of numerical algorithms are available.

The solution of the Hamilton-Jacobi-Bellman partial differential equation requires an amount of storage space (and computation time) that grows exponentially with the number of dimensions of the state space (the so-called "curse of dimensionality"), and so is feasible only for systems with a state space of a "small" dimension.

Since the optimal control approach is not practically feasible for real-time control of non-trivial systems , other tools and ideas have been introduced, which can better exploit the structure of the problem.

Differential Flatness

Differential flatness is a property enjoyed by some mechanical systems which was first introduced and studied by Fliess, Levine, Martin and Rouchon [35, 112], and which results in very efficient methods for steering nonlinear systems in real-time [137].

Writing the system dynamics as $\dot{x} = f(x,u)$ (where the dot represents differentiation with respect to time), we say that the system is differentially flat if it is possible to find a set of flat outputs $z = \zeta(x, u, \dot{u}, \ldots, u^{(l)})$, $z \in \mathbb{R}^m$ (where m is the number of independent control inputs), such that it is possible to recover the complete state and control trajectory from *any* trajectory of the flat outputs, and a *finite* number of its derivatives:

$$\begin{aligned} x &= x(z, \dot{z}, \ddot{z}, \ldots, z^l), \\ u &= u(z, \dot{z}, \ddot{z}, \ldots, z^l). \end{aligned}$$

If a system is differentially flat, then the steering problem can be solved by finding values of the flat outputs at the initial and final conditions, connecting them with a time-parameterized, sufficiently smooth curve, and then recovering the control input history. Differentially flat systems include controllable linear systems, and nonlinear systems which are feedback linearizable (either by static or dynamic feedback [49]), and as such include a very large set of mechanical systems of interest, including approximations of aircraft [90, 89, 41]

and rotorcraft [138, 62]. Unfortunately though, no general method is available yet to ascertain differential flatness of a given system.

Moreover, even for differentially flat systems, currently there is no straightforward way of taking into account the flight envelope constraints (1.1). Research in this direction is currently very active; computationally efficient but possibly conservative methods for generating feasible trajectories for differentially flat systems have been introduced very recently in [34, 94].

Other nonlinear control methods

The nonlinear control community has, in the recent years, produced several other techniques for addressing the problem of steering nonholonomic systems. We will mention and provide pointers to the reference material for just a few of the main methods; a review and a discussion of most of the following techniques is available in the nonlinear control textbook by Sastry [114]. The method of steering by sinusoidal inputs, at integrally related frequencies, has been introduced by Murray and Sastry to steer systems which are feedback-equivalent to chained form [96]. Small-amplitude sinusoidal forcing was also used by Bullo, Leonard *et al.* for steering underactuated Lagrangian systems on Lie groups[24]. Laferriere and Sussmann developed a general algorithm based on the application of constant inputs and the Campbell-Baker-Haussdorf formula [66]; the algorithm is exact for nilpotent systems, and provides approximate steering for general systems. Monaco and Normand-Cyrot apply multi-rate control for systems that admit an exact sampled model [95].

While all these methods provide valuable insights and a sound theoretical basis for steering of nonholonomic systems, their application is based on assumptions which are not always satisfied. In fact, many mechanical systems of interest are *not* Lagrangian, differentially flat, nor nilpotent, and cannot be reduced to strict-feedback or chained form. Moreover, the above mentioned techniques do not generally take into account limitations and constraints deriving from the physical implementation of the controlled system. Actuator position and rate saturation, and constraints on the states (such as those arising from the necessity to keep the system in "safe" operating conditions) are not generally directly addressed. Finally, these methods concentrate on the construction of feasible trajectories between two points, regardless of the "quality" of the resulting motion plan, in terms of cost functionals of the form (1.2).

As a matter of fact, the problem of stabilization and control of general underactuated mechanical systems is still considered a major open problem [128, 110].

Steering by Control Quanta

A fundamentally different idea, namely motion planning through *"control quanta"* was recently introduced by Marigo and Bicchi [84]. Even though independently developed, the control quanta method is based on some of the same principles behind the control framework that will be developed in this dissertation.

The control quanta method is applicable to driftless control systems which can be written in strictly triangular form, and provides approximate (yet to an arbitrary precision) steering of the system to the goal configuration. A driftless system is said to be in strictly (block)

triangular form (ST) if its dynamics can be represented as follows:

$$\begin{aligned} \dot{x}_1 &= f_1(x_2, \ldots, x_p)^T u \\ \dot{x}_2 &= f_2(x_3, \ldots, x_p)^T u \\ &\vdots \\ \dot{x}_{p-1} &= f_{n-1}(x_p)^T u \\ \dot{x}_p &= u \end{aligned} \tag{1.9}$$

where $x = [x_1, x_2, \ldots, x_p]^T \in \mathbb{R}^{n_1 \times n_2 \times \ldots \times n_p} = \mathbb{R}^n$, and $u \in \mathbb{R}^{n_p}$. The ST form represents a very large class of nonlinear driftless systems; in particular it includes systems in chained form and nilpotent systems. Notice that the system (1.9) is characterized by a "cascade" of symmetry groups, in the sense that the dynamics of $x_{i+1}, \ldots, x_p$ are invariant with respect to translations on $x_1, \ldots, x_i$, for $i < p$ (more on symmetries in Chapter 2).

A *control quantum* is defined as a function $u^q : [0, T^q] \to \mathcal{U}$ (where the superscript is used to indicate the "index" of each control quantum), such that the execution of u^q over a time interval $[0, T^q]$ leaves the state components $x_{i+1}, \ldots, x_p$ unchanged. On the other hand, the execution of u^q will result in a net vector displacement (drift) Δ^q for the states $x_1, \ldots, x_i$. Given a collection of control quanta $u^1, u^2, \ldots, u^l$, through a repeated application of control quanta the system will be moved on $\mathbb{R}^{n_1 \times n_2 \times \ldots \times n_i}$, and the reachable set will have the structure of a lattice. It has been proven in [86] that, depending on the selection of control quanta, the reachable set (for $x_1, \ldots, x_i$) will either be everywhere dense, or nowhere dense.

To fix ideas, consider the following simple example, in which two control quanta are used to control the system known as the Brockett's nonholonomic integrator.

Example 1.1 (Control Quanta) *Consider the system:*

$$\begin{aligned} \dot{y} &= x_1 u_2 - x_2 u_1 \\ \dot{x}_1 &= u_1 \\ \dot{x}_2 &= u_2. \end{aligned}$$

Consider the following set of control quanta (for a fixed $k > 0$):

$$\begin{aligned} u^1(t) &= [\sin(t), \cos(t)]^T \\ u^2(t) &= \sqrt{k}[\sin(t), -\cos(t)]^T \end{aligned}$$

over the interval $[0, 2\pi]$.

We want to analyze the reachable set from zero initial conditions. The execution of one control quantum results in no net displacement for x_1 and x_2, but causes y to change by the following amount:

$$\begin{aligned} \Delta^1 &= 2\pi \\ \Delta^2 &= -2k\pi. \end{aligned}$$

The reachable set (for y) is then nowhere dense if k is rational, everywhere dense otherwise.

In the case in which the reachable set is everywhere dense, the control quanta method does provide approximate steering to arbitrary precision. However, the time required to get arbitrarily close to a given point can grow to infinity.

The method that we will present in this dissertation is applicable to a more general class of nonlinear systems, including systems with drift. Moreover, by exploiting the additional degree of "slackness" provided by natural trajectories of the system (the relative equilibria, to be defined in Chapter 2), we can provide *exact steering in finite time*. However, if the system does not admit non-trivial relative equilibria, the framework that will be presented in the following reduces to a version of the control quanta methodology.

In this dissertation, we will provide a framework and algorithms for autonomous vehicle motion planning that are applicable to all time-invariant dynamical systems which admit dynamic symmetries and the associated natural trajectories, or relative equilibria (to be defined). No other assumptions are made on the dynamics, thus resulting in an exact motion planning technique of general applicability. Under suitable assumptions, the technique results in almost optimal trajectories, and satisfies actuator and state constraints by construction. Building on a relatively expensive off-line computation phase, we provide a motion planning algorithm that is viable for real-time practical applications.

1.1.3 Motion planning with obstacles

The nonlinear control tools for steering nonholonomic systems do provide a sound theoretical basis for motion planning, but do not take into account "external" constraints, such as obstacles in the environment. The problem of planning a trajectory in an environment cluttered by obstacles has been the object of considerable interest in the robotics and artificial intelligence community; most of the activity has focused on holonomic, kinematic motion problems (i.e. Problem 1.4, with no differential constraints).

Motion planning research is a well-established discipline in robotics [68, 67, 74, 70]. Roughly speaking, it is possible to identify three general approaches to the motion planning problem, namely: cell decomposition methods, roadmap methods, and artificial potential field methods [68].

The cell decomposition methods rely on the partition of the configuration space* into a finite number of regions, in each of which collision-free paths can be found "easily". The motion planning problem then is translated into the problem of finding a sequence of neighboring cells, including the initial and final conditions [76].

In roadmap methods, a network of collision-free connecting paths is constructed, which spans the free configuration space. The path planning problem then reduces to finding paths connecting the initial and final configuration to the roadmap, and then selecting a sequence of paths on the roadmap. There are several methods for building such a roadmap, among which we can mention visibility graphs [77, 32] and Voronoi diagrams [99].

Finally, in artificial potential field methods, a collision-free trajectory is generated by the robot moving locally according to "forces" defined as the negative gradient of a potential function [58, 48, 8]. This function is designed to provide attractive forces towards the goal, and repulsive forces which push the robot away from obstacles (the potential function is bowl-shaped with the goal at the bottom, and obstacles are represented by peaks). Notice that this class of methods is based on the definition of a feedback control policy (i.e. the control is computed at each instant in time as a function of the current state), as opposed

*A configuration of a robot identifies the position of all of its points with respect to an inertial reference frame. We will assume that such a configuration can be described by a finite number of parameters. The *configuration space* is the set of all possible configurations of the robot. If the system is purely kinematic, the state space and the configuration space are equivalent. See Chapter 2.

to the open-loop approach of the previous two classes. The problem with this formulation is the possible existence of local minima, in which the robot might become trapped. An artificial potential function which does not have local minima is said a navigation function, but computing such a function is in the general case as difficult as solving the motion planning problem for all initial conditions [111]. As a matter of fact, notice that the optimal cost-to-go function $J^*(x)$ (defined in Section 1.1.2) is a navigation function.

The algorithms for motion planning must be evaluated in terms of completeness and computational complexity. An algorithm is said to be *complete* if it returns a valid solution to the motion planning problem if one exists, and returns failure if and only if the problem is not feasible: this is what we will call a correct termination for a motion planning algorithm. The computational complexity of some basic formulations of the motion planning problem has been studied in detail. The so-called "generalized mover's problem", involving path planning[†] for robots made of several polyhedral parts, among polyhedral obstacles has been proven by Reif [109] to be PSPACE-hard[‡]. An algorithm for path planning in a configuration space of dimension n, with obstacles defined by m polynomial constraints of degree d has been established by Schwartz and Sharir [116]; the time complexity of the algorithm is (twice) exponential in n, and polynomial in m ("geometrical complexity") and in d ("algebraic complexity"). The most efficient algorithm currently available for solving the same problem, due to Canny [27], is singly exponential in n. The above results strongly suggest that the complexity of the path planning problem grows exponentially in the dimension of the configuration space.

Kinodynamic Motion Planning

Moreover, kinematic, holonomic path planning is not enough for many problems of interest, particularly problems involving "agile" autonomous vehicles, for which we have to take into account the additional constraints on the vehicle's motion arising from its dynamics, or from non-holonomic constraints (that is, non-integrable constraints on the state and its derivatives).

Even though in some cases it is possible to force a dynamical system to follow trajectories generated by a kinematic model, this is not true in general. Even in the case in which this is possible (explicit conditions are given in [23]) , forcing the system to follow an arbitrary trajectory could result in very slow trajectories. For example, a helicopter can, in principle, fly along a piecewise-linear trajectory, but it has to stop at every corner. Similar arguments arise in the presence of non-holonomic constraints: for some non-holonomic systems it is possible to approximate arbitrary paths, but usually this requires the execution of complex sequences of small-amplitude movements about the reference path (e.g. [69, 74]; see also the references on steering of nonlinear systems), which could possibly result in a large loss in terms of performance of the final trajectory.

If, on the other hand, the output of a kinematic planner is used as a reference trajectory for an "inner loop" consisting of a tracking control law, the discrepancies between the planned trajectory and the trajectory actually executed by the system can be relevant, and lead to collision with obstacle even in the case in which the planned trajectory was

[†]Here we distinguish between *motion planning* and *path planning*; by path planning we mean the generation of collision-free trajectories without taking into account dynamics or non-holonomic constraints.

[‡]The complexity class PSPACE includes decision problems for which answers can be found with resources (memory) which are polynomial in the size of the input. The run time is not constrained. The complexity class NP is known to be a subset of PSPACE; moreover, it is believed to be a *proper* subset [117].

collision-free. This is true, for example, in cases in which the characteristic dimensions of the environment (e.g. the size of the obstacles, and of the gaps between them) and of the vehicle's dynamics (e.g. the turning radius at some nominal velocity) are comparable.

As a consequence, it is desirable that motion planning strategies take fully into account the dynamics of the vehicle. In other words, it is desirable that the output from the motion planning be executable by the system dynamics: this is the object of a recent direction in motion planning research, usually referred to as kinodynamic motion planning.

Clearly, such a problem is at least as difficult as the path planning problem. Moreover, constraints deriving from the system's dynamics, or from non-holonomic constraints cannot be represented as "forbidden zones" in the state space (i.e. the space encoding the configuration of the vehicle, as well as its velocities). There is strong evidence that any deterministic and complete algorithm for the solution of kinodynamic motion planning problems will require at least exponential time in the dimension of the state space of the dynamical system, which is usually at least twice the dimension of its configuration space. As a consequence, available algorithms are implementable in practice only for systems of very small dimension. Since the state space of aerospace vehicles is at least 12 (when vehicles are modelled as rigid bodies), one has to resort to heuristic techniques, or seek alternative formulations of the problem.

Randomized Motion Planning Algorithms

In order to circumvent the computational complexity of deterministic, complete algorithms, a new class of motion planning algorithms, known as Probabilistic RoadMap (PRM) planners, was introduced by Kavraki, Kolountzakis, and Latombe [54], and Svestka and Overmars [101]. These algorithms were subsequently refined in [57, 55, 44, 47]. The PRM path planning architecture was first introduced as a fast and efficient algorithm for geometric, multiple-query path planning. The original PRM planner is based on an off-line pre-processing phase and an on-line query phase. The pre-processing phase is aimed at constructing a graph of feasible paths in the entire configuration space (the *roadmap*), which would make future queries "easy" to solve. The on-line query phase selects an appropriate path from the already computed roadmap, together with the computation of two "short" paths to connect starting and ending points to the closest nodes (or *milestones*) of the roadmap. The PRM algorithm has been proven to be complete in a probabilistic sense, that is, the probability of correct termination approaches unity as the number of milestones increases. Moreover, performance bounds have been derived as a function of certain characteristics of the environment (i.e. its *expansiveness*, tied to the rate at which the set of points which can be connected to the roadmap grows with the size of the roadmap) which prove that the probability of correct termination approaches one exponentially fast in the number of milestones [46].

However, for many path planning applications, building a roadmap *a priori* may not be required, or even feasible (e.g. for planning in a dynamic, rapidly changing environment). In addition, the basic roadmap algorithms neglect the vehicle dynamics: the vehicle usually needs to navigate along a piecewise linear trajectory with very sharp turning points. To address both points, a variant on the initial randomized motion planning algorithm was developed by introducing the concept of Rapidly-exploring Random Trees (RRTs) [71, 72, 65]. The RRT algorithm consists of building on-line a tree of feasible trajectories by extending branches towards randomly generated target points. While in the PRM approach the idea was to explore the configuration space exhaustively in the pre-processing phase, RRTs tend

to achieve fast and efficient single-query planning by exploring the environment as little as possible. A significant feature of the RRTs is that the resulting trajectories are by definition executable by the underlying dynamical system. In addition, under appropriate technical conditions, the RRT algorithm has been proven probabilistically complete [65], that is, the probability of finding a path from origin to destination converges to one if such a feasible path exists.

In Hsu *et al.* [45] a new incremental roadmap building algorithm is introduced that provides not only probabilistic completeness, but also recovers performance guarantees on the algorithm. That is, the probability of the algorithm finding a solution if one exists converges to one exponentially fast with the number of random samples used to build the tree. One difficulty pointed out by the authors, however, lies with the fact that algorithm performance relies upon uniform sampling of milestones in the reachable space of the tree. Practical implementations (which have reportedly demonstrated excellent performance [45]) rely upon uniform sampling of the system's control inputs instead, which in general does not guarantee uniform sampling of the workspace.

Motivated by these recent developments, a new randomized, incremental path planning algorithm is proposed in this dissertation. This incremental roadmap building algorithm is able to effectively deal with the system's dynamics, in an environment characterized by moving obstacles. Central to this algorithm is the assumption that an *obstacle-free* guidance loop is available, which is able to steer the system from any state (including configuration and velocity) to any desired configuration at rest, assuming that there are no obstacles in the environment. This guidance loop enables uniform sampling of the workspace while generating trajectories that are executable by the dynamical system. As a consequence, it is shown that this path planning algorithm satisfies the technical conditions elicited by Hsu *et al.* [45] and therefore offers guaranteed performance, in the form of bounds on the convergence rate to unity of the probability of correct termination. Considering real-time computation issues, the path planning algorithm provides *safety* guarantees in the sense that it provides intermediate milestones with guaranteed buffer time before a collision occurs. If possible and practical, the buffer time can be extended to infinity, thus resulting, in principle, in hard safety guarantees on the generated plan. Moreover, in the case in which the obstacle-free planner is optimal with respect to some meaningful cost, it is possible to implement branch-and-bound algorithms to bias the search for a solution in order to maximize the performance of the computed solution.

1.2 Hierarchical systems

As we have just seen, general motion planning problems can be split up into two sub-problems that are themselves considered to be extremely difficult: steering of an underactuated mechanical system and generation of an obstacle-free path. The level of complexity of the motion planning problem will thus be daunting in the general case. A common way of dealing with highly complex systems — not necessarily autonomous vehicles — is based on a hierarchical decomposition of the problem, into a sequence of problems that can be solved more easily (the "divide and conquer" principle). Consequently, this will result in the introduction of a hierarchy of control and decision layers, each of them responsible for the solution of a sub-problem[§].

[§]As a matter of fact, recall that the basic assumption for our own motion planning algorithm is the existence of an obstacle free guidance loop: we have already introduced a hierarchical structure.

In the context of autonomous flying vehicles, Stengel [129] identifies three main layers in the hierarchy, namely:

- **Strategic layer:** This layer includes a number of *declarative functions*, which are responsible to assess the current status of the system and the operational scenario, with the goal of defining, and prioritizing, the mission objectives. Task scheduling and generation of way-points are examples of activities in this layer.

- **Tactical layer:** This layer includes *procedural functions* which aim at devising plans to achieve the goals set by the strategic layers using the available resources. The traditional functions of guidance, navigation, and possibly adaptation and reconfiguration can be included in this layer.

- **Skill layer:** This layer includes all the *reflexive functions* performed by the control system, i.e. all of those actions which an expert pilot would perform without a conscious effort, to execute the plan devised by the tactical layer. At this level the control system interacts directly with the physical plant (the vehicle). Activities traditionally defined as control, regulation and estimation are included in this layer.

Such a decomposition is attractive from a conceptual point of view, and provides the means for achieving computational tractability for very complex problems: systems based on such a decomposition have been successfully developed, and in general perform satisfactorily. However, there are some shortcomings which are inherent to the introduction of arbitrary hierarchical structures, which relate to the difficulty of the formal analysis and verification of the complete system.

First of all, the stability and the performance of the interconnection of several layers of dynamical systems are difficult to prove in the general case. Usually, arguments based on time-scale separation are invoked, which aim at decoupling the dynamics of the different layers. However, and especially when high levels of performance are desired, such a separation might not exist (or, conversely, artificial conservative limits have to be imposed on the achievable performance levels). Also, if the system must be able to react to changes in a dynamic and uncertain environment, this assumption is likely to be violated.

Another source of possible problems is the lack of hierarchical consistency. In general, commands that are generated by one of the hierarchical layers might not be executable by the lower layers. In this case, the higher layers in the hierarchy expect too much from the lower layers. The mismatch between commands and actual implementation can lead to undesirable behaviors, such as collision with obstacles, or instability of the overall closed-loop system. Also in this case, the problem becomes more pressing if high levels of performance are desired. Current research work aims at establishing conditions under which hierarchical abstractions of continuous systems are consistent, in the sense that all behaviors commanded by a hierarchical layer are actually executable by lower layers [103, 104].

One of the key ideas in this dissertation is the construction of a consistent hierarchical structure that respects the natural dynamics of the system. Instead of forcing an arbitrary hierarchical structure -- which will be difficult to analyze and verify — a consistent hierarchy, and hence a consistent abstraction, is constructed "from the ground up", exploiting the fundamental properties of the system's dynamics.

1.2.1 Hybrid Hierarchical Systems

The recent increase in computational power and storage capacity, not only for desktop computers, but also for embedded processors, motivates an ever-increasing interest in the development of software-driven systems, including autonomous vehicles. While digital computers have been used for a few decades to perform the functions associated with the skill layer, namely regulation/control and estimation, and some limited forms of guidance and navigation, now the capabilities exist to expand the role of software and computers to the higher hierarchical layers. The interaction that occurs at the highest abstraction levels between physical, continuous systems, and software-driven systems, creates a new set of challenging problems in systems and control theory.

In particular, we are concerned with dynamical systems which evolve on a state space that includes both a continuous and a discrete component. These systems are commonly referred to as *hybrid systems* in the recent literature [3, 18, 79].

In our case, the dynamics of the physical plant (e.g. an autonomous vehicle) are inherently continuous, and as such are described by ordinary differential equations. Even though digital computers are typically used for control purposes, very often the software encoding the reflexive functions runs at a very high rate, and the skill layer is also modelled as a continuous system. Even in the cases in which it is required to model the skill layer as a discrete-time system, the effects of the finite word length are generally neglected, and the state space is considered to be continuous. However, higher control functions, such as those included in the "strategic" layer (e.g. task scheduling, goal selection, etc.), are implemented by logical decision-making agents, usually operating on a discrete set of state and decision variables. Often, even some functions of the procedural layer, such as guidance, include discrete elements, represented for example by mode switches between different operating conditions (e.g. cruise, land, take off, hover, etc.).

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 [18, 79, 82] and references therein). General hybrid systems, derived from arbitrary hierarchical decompositions, however, can be extremely difficult to analyze and verify, and results can be obtained only for limited classes of systems [2, 43, 40].

On the other hand, in many cases of interest, it is convenient to keep in mind that the "hybridness" of the system derives from the *design* of the control system, and is not inherent to the plant to be controlled[¶]. In other words, the introduction of discrete dynamics, or logic, in the controller implementation is a *design choice.* In all these cases, it is convenient to exploit the degrees of freedom available in the control design process to build a hybrid system in such a way that it offers safety and performance guarantees *by design*, at least in an idealized situation, as defined by a nominal or clearly identified model of the vehicle and its environment. In other words, instead of imposing an arbitrary hierarchical structure on the system, which could result in a very difficult analysis problem, it is desirable to exploit the designer's insight into the nominal behavior of the system. Consequently, the analysis and verification problems of the hybrid system are translated to a robustness analysis problem, which can be addressed using the relevant tools from systems and control theory [29, 140, 16, 80, 136, 81].

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,

[¶]Naturally, there are many notable and important exceptions, as in the case of legged robots, or systems including switches and other hard nonlinearities

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 [60, 61, 118]. The formal verification of this kind of hybrid systems can be extremely hard, and is often foregone in favor of simulations [75]. Other hybrid system architectures proposed for air traffic conflict resolution have a rigorous mathematical foundation, but do not always provide feasible trajectories for the system [136, 134], 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 [91, 137], maintaining the vehicle inside an assigned flight envelope [135, 100].

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.

1.3 Maneuver Automaton

In much of the aerospace literature, the expression *"maneuver"* is often mentioned, and used to refer to some building blocks for solving guidance and control problems [120, 7, 124, 41, 132, 17]. Indeed, the analysis of human-piloted flight test data shows that experienced pilots fly by interconnecting consistent and repeatable "maneuvers". Flight test data were collected from an instrumented radio-controlled helicopter, during the execution of acrobatic maneuvers. In figure 1-1, adapted from [36], the state and control histories recorded during the execution of "barrel rolls"[‖] maneuvers are reported: as it can be seen successive executions of the barrel roll maneuver resemble one another very closely.

Moreover, humans tend to think of trajectories by means of aggregate notions, such as specific "maneuvers" and not as the detailed time-histories of position, velocity, and angular rates. Indeed, it can be appealing to implement a control framework in which both the state and the decision variables are based on such aggregate notions: for example, the state can be described as "flying straight and level at 50 knots", and a possible decision could be the request to "establish a steady turn rate of 30 degrees per second to the right", or even to "execute a loop" (and this is in short what we will do in Chapter 3).

However, to the author's knowledge, there is no formal definition of maneuver in the literature; in particular, there is no definition of maneuver that allows the construction of a framework for mathematical programming, and consequently the computation of "optimal" sequences of maneuvers, with respect to some well-defined cost. One of the main contributions of this work is a formalization of the concept of maneuver, and the introduction of a

[‖]During a barrel roll maneuver, the vehicles rolls about its velocity vector, resembling the motion inside a horizontal barrel

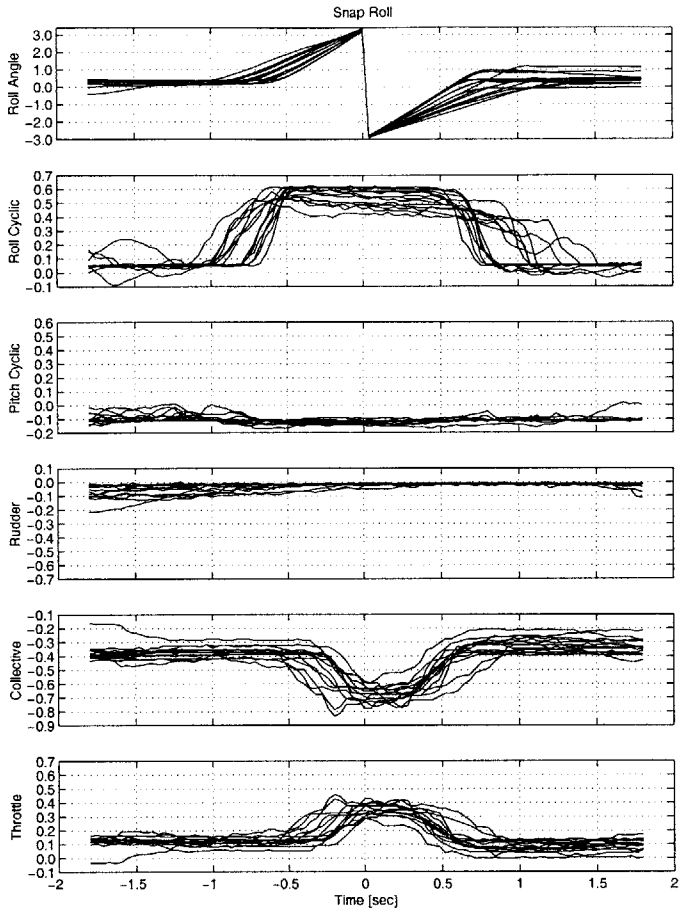

Figure 1-1: Maneuvers are often repeatable.

maneuver-based framework for the solution of optimal control problems. Inspired by the above considerations, 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. By quantization of the system dynamics we mean that that we restrict the feasible nominal system trajectories to the family of time-parameterized curves that can be obtained by the interconnection of appropriately defined primitives.

These primitives then constitute a "maneuver library" from which the nominal trajectory will be constructed. At the core of the control architecture lies a hybrid *Maneuver Automaton*, the states of which represent feasible trajectory primitives for the vehicle. Each constituent subsystem of the hybrid controller will be 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 observed 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.

1.4 Statement of contributions

The main contribution of this dissertation is the development and the analysis of the Maneuver Automaton, which is proposed as a new modelling tool for planning the motion of autonomous vehicles. By implementing the full, continuous dynamics of the vehicle, (usually described by a set of Ordinary Differential Equations), as a hybrid control system, we are able to achieve the following:

- A compact representation of an important subset of the system's dynamics, by which all the relevant information about the state of the vehicle is encoded by a small set of parameters. This representation is instrumental in reducing the computational complexity of motion planning, but also can be seen as the result of a "model reduction" exercise, aimed for example at increasing the semantic value of the state information. This can be helpful in addressing issues like multiple vehicle coordination with limited communication resources.

- A consistent and asynchronous hierarchical decomposition of the system: all reference trajectories generated by the maneuver automaton are by construction executable by the physical system. Moreover, the rate at which decisions are made at the level of the maneuver automaton is not directly related to the natural frequencies of the physical system, or to the bandwidth of the "low-level" feedback control loop. Consequently, what we achieve is the first truly integrated, hierarchically consistent guidance and control architecture for aerospace and other vehicles available in the literature. With such an architecture high-level concerns like safety, and obstacle avoidance and ultimately mission success, and low-level concerns like stability, are consistently dealt with, within the same framework.

- A computationally tractable formulation of optimal control problems, even for high-dimensional systems. We show that the motion planning problem can be reduced to a control problem over a reduced space, which nonetheless encodes all the relevant information about the system.

- A formulation exploiting some of the natural, fundamental properties of the dynamical systems of interest (symmetries). Since the only assumptions on the system's dynamics rely on these properties, the formulation we propose does not require the introduction of approximations or simplifications on the dynamics model (at least in a nominal situation).

- A formulation which is by nature amenable to both model-based control design and experiment (flight-test) based design; this is particularly useful for aerospace systems with complex dynamics, for which the development of a very accurate model is unfeasible or possibly even undesirable.

The advantages listed above are obtained at the expense of imposing additional constraints on the vehicle dynamics or, in other words, on the reference trajectory generation process. These additional constraints come from the fact that the reference nominal trajectories are constrained to be the output of the hybrid system encoding the maneuver automaton. When addressing optimal control problems, and motion planning issues, we run into respectively sub-optimality and path-incompleteness (with respect to the notion of completeness in Section 1.1.3). However, we provide some insight about the following questions:

- What is the optimal quantization, that is, what is the optimal choice of the set of trajectory primitives in the maneuver automaton, which will result in the least penalty on the optimal achievable performance?
- Conversely, given a choice of trajectory primitives, and the specification of a maneuver automaton, what is the resulting loss in achievable performance?

One of the problems we have to address in the construction of a Maneuver Automaton is the design of appropriate maneuver tracking feedback control laws. While there are many control law design methods in the literature that are widely used for aerospace vehicles, there are none which are able to track all trajectories of an air vehicle in a seamless fashion, or without introducing ad-hoc solutions. Most control design methods for aircraft trajectory tracking rely on some minimal parameterization of the configuration of the vehicle. In such cases, tracking acrobatic maneuvers, in which an air vehicle undergoes large attitude variations, can be made impossible or cumbersome because of artificial singularities, induced by the choice of the parameterization. We present a control design method for aggressive maneuver tracking for small helicopters, which, by operating directly on the configuration manifold of the vehicle (the group of rigid motions in the three-dimensional space) achieves the following:

- Asymptotic (locally exponential) tracking of all feasible trajectories.
- Bounded tracking in the presence of bounded disturbances.
- An invariant formulation of the control laws exploits the natural symmetries of the system, and results in constant gains in body axes.

The final part of the dissertation is devoted to the development of motion planning algorithms in a dynamic environment. Arguably the most exciting advances in the motion planning literature have come in the recent years from the introduction of randomized motion planning algorithms, briefly described in Section 1.1.3. The development of convergence and performance guarantees of these algorithms depends on uniform sampling of the workspace, something which was devoted to an idealized procedure in the literature [45], but was dismissed in favor of uniform sampling in the input space in practical implementations. By employing the optimal control solution in a free space developed for the maneuver automaton (as well as any other optimal control solution for a general system), we show:

- How to implement a randomized algorithm with uniform sampling of the reachable space, through on-line scheduling of (possibly optimal) control policies.
- How to use the optimal control solution in a free environemnt to guide the search for an optimal motion planning solution through branch-and-bound arguments.
- How to ensure safety in a dynamic environment, in the face of finite computation time.

1.5 Outline

This dissertation is organized as follows. In Chapter 2 we will present some background material on mechanical control systems, and will present models for the dynamics of some systems and vehicles of interest.

In Chapter 3 we will introduce the Maneuver Automaton framework, and will analyze its properties. Moreover, we will address the problem of optimal control (and hence of motion planning) in an obstacle-free environment.

In Chapter 4 we will address the problem of building a Robust Maneuver Automaton for a system with disturbances and/or uncertainties in the plant or in the environment. In particular, in Chapter 5, we will provide a control design method that is applicable to aggressive maneuvering of small autonomous helicopters.

In Chapter 6 we will address the problem of real-time motion planning in a dynamic environment (e.g. an environment characterized by moving obstacles). Finally, in Chapter 7 we will draw some conclusions, and present some ideas for future research in the area.

Chapter 2

Models of Mechanical Control Systems

This chapter provides some background on mechanical control systems. We will provide models of various classes of mechanical systems, with a particular emphasis on models of vehicles for which motion planning algorithms will be examined in the dissertation. The key idea is that mechanical control systems are a very structured class of nonlinear control systems, arising from principles of geometric mechanics [4, 87]. As a consequence, mechanical control systems often enjoy some properties that can simplify the design of control laws.

The presentation of the models will be based on a Lagrangian mechanics formulation; for systems whose configuration belongs to a Lie group, we will also give the corresponding dynamics formulation. First, we will introduce some background and basic concepts of Lagrangian mechanics. Second, we will extend these concepts to the case in which the system admits a Lie group representation. Third, we will define and examine one of the fundamental properties of mechanical systems (symmetry), which will be at the basis of the motion planning framework that will be developed throughout this dissertation. Last, we will proceed to the analysis of some mechanical systems that will serve as application examples, ranging from simple academic examples and laboratory setups, to detailed models of small autonomous helicopters.

2.1 Mechanical Control Systems

In this section we introduce a general framework for describing the dynamics of mechanical control systems, based on the Lagrangian mechanics formulation. Complete expositions can be found in [1, 87, 115].

Given a mechanical system, we need to identify uniquely the position in an inertial reference frame of all its particles. We will assume that the position of all particles of mechanical systems of interest can be determined via an element of a finite-dimensional manifold Q, that we will call *configuration manifold*, or *configuration space*. Any element $q \in Q$ will define a *configuration* of the mechanical system. If the manifold Q is locally diffeomorphic to an n-dimensional Euclidean space $\mathbb{R}^n$, then we can define a coordinate chart as a local diffeomorphism $\chi : Q \to \mathbb{R}^n$ which associates to an element of the manifold a set of n real numbers (the coordinates).

If we denote by $C^\infty(Q)$ the set of all smooth real-valued functions on Q, then we can

define the *tangent space* T_qQ to the manifold Q at the configuration q as the set of all derivations on $C^\infty(Q)$ at q. The tangent space T_qQ is an n-dimensional linear space, and its elements are *tangent vectors*. The *tangent bundle* TQ is defined as the union over all $q \in Q$ of the tangent spaces T_qQ; the tangent bundle is a smooth manifold of dimension $2n$, and each of its elements is characterized by two entries (q, v), where $q \in Q$, $v \in T_qQ$.

The set of all linear functions $T_q \to \mathbb{R}$ is itself a linear space, known as the dual of T_qQ, or cotangent space, and is denoted by T_q^*Q. The cotangent bundle T^*Q is defined as the union over all $q \in Q$ of the cotangent spaces T_q^*Q. We can think of a vector $v \in T_qQ$ as velocity, and of a covector $p \in T^*Q$ as momentum; the natural pairing $\langle p, v \rangle$ then corresponds to energy; alternatively, elements of T_q^*Q can be thought as forces, and the pairing with velocities corresponds to power.

Formally, the kinetic energy of a mechanical system is a Riemannian metric on Q, that is a smooth map that associates to each tangent space an inner product $\langle\!\langle \cdot, \cdot \rangle\!\rangle_q$; correspondingly, we can define the inertia tensor $\mathbb{M}_q$ as a mapping from T_qQ to its dual, so that $\langle\!\langle v, v \rangle\!\rangle_q = \langle \mathbb{M}_q(v), v \rangle$. In coordinates, if velocities are represented by vectors in $\mathbb{R}^n$, the inertia tensor can be represented by a positive-definite $n \times n$ matrix $M(q)$.

Following [22], we define a mechanical control system in the following way:

Definition 2.1 (Mechanical Control Systems) *A mechanical control system is defined by the following objects:*

1. *an n-dimensional manifold Q, the* configuration manifold, *with local coordinates $q = \{q_1, q_2, \ldots, q_n\}$.*
2. *A Riemannian metric $M_q : TQ \times TQ \to \mathbb{R}$ on Q, describing the kinetic energy.*
3. *A function $V : Q \to \mathbb{R}$, describing the potential energy.*
4. *A map $B : TQ \to T^*Q$, describing the action of non-conservative forces on the mechanical system.*
5. *An $(n_u + n_w)$-dimensional codistribution $\mathbf{F} = \operatorname{span}\{F^1(q), F^2(q), \ldots, F^{(n_u+n_w)}(q)\}$, defining the input directions.*
6. *A compact input set $\mathcal{U} \times \mathcal{W} \subset \mathbb{R}^{n_u} \times \mathbb{R}^{n_w}$. The control input are partitioned into a set of control inputs $u \in \mathcal{U}$, and a set of disturbance inputs $w \in \mathcal{W}$[†].*

The Lagrangian of the system can be written as:

$$L(q, \dot{q}) = \frac{1}{2} \langle\!\langle \dot{q}, \dot{q} \rangle\!\rangle_q - V(q).$$

Equivalently, assuming that the Riemann metric on Q can be represented with the positive definite inertia matrix $M(q) \in \mathbb{R}^{n \times n}$, we have

$$L(q, \dot{q}) = \frac{1}{2} \dot{q}^T M(q) \dot{q} - V(q). \tag{2.1}$$

The Euler-Lagrange equations, which describe the dynamics of the system, will then be given by [4]

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}} - \frac{\partial L}{\partial q} = B(q, \dot{q}) + \sum_{i=1}^{m} F(q)^i u_i. \tag{2.2}$$

[†]In the following, to simplify notation, we will assume $n_w = 0$ and $\mathcal{W} = \emptyset$, unless otherwise specified (as in Chapter 4)

In coordinates, the above equations are written as follows:

$$\ddot{q}_i + \Gamma^i_{jk}\dot{q}_j\dot{q}_k = M^{-1}_{ij}\left(-\frac{\partial V}{\partial q_j} + B_j(q,\dot{q}) + \sum_{k=1}^{m} F^k_j u_k\right)$$

where the *Christoffel symbols* Γ^k_{ij} are defined as:

$$\Gamma^k_{ij} := \frac{1}{2}\sum_m (M^{-1})_{mk}\left(\frac{\partial M_{mj}}{\partial q_i} + \frac{\partial M_{mi}}{\partial q_j} + \frac{\partial M_{ij}}{\partial q_m}\right).$$

For simple Lagrangian systems $B(q,\dot{q})$ is identically zero. However, since many mechanical systems of interest, especially vehicles, are subject to significant non-conservative forces and moments, such as those due to friction, or air drag, we need to maintain the capability to deal with these generalized forces.

2.1.1 Underactuation and nonholonomy

A system is said *fully actuated* if the dimension of the codistribution $\mathbb{F}$, that is the number of independent input directions, is the same as the dimension of the configuration manifold Q. Accordingly, a system is said *underactuated* if the number of independent control inputs is smaller than the dimension of the configuration manifold Q.

If a system is fully actuated, it can be easily steered along any given curve on the configuration manifold, i.e. it is *controllable*. Feedback linearization [49], or other techniques can be used to derive in a relatively easy way a control that steers the system exactly along the desired trajectory. This is not true in general for an underactuated system. However, an underactuated system can be locally controllable if it enjoys the property of non-holonomy [114]. A (second order) nonholonomic constraint is a non-integrable constraint of the form $\nu(q,\dot{q},\ddot{q}) = 0$. The existence of nonholonomic constraints translates into the fact that the system can be locally steered along a manifold of dimension larger than the number of independent control inputs. If the degree of non-holonomy of the system is sufficiently large, the system is indeed locally controllable (Chow's theorem). In this case it will be possible to track any curve on the configuration manifold.

Thus, while non-holonomy is very often seen as a nuisance, since it complicates the process of generating feasible trajectories for the system (and makes parallel parking difficult), from an engineering perspective non-holonomy is a beneficial property to be exploited, since it allows the reduction of the number of actuators on the system, or vehicle. Even though parallel parking can be difficult for a novice driver, it allows engineers to design simpler vehicles, without redundant actuators: If the rear wheels could steer, parking would be easier, but the vehicle would be unnecessarily complex. Interesting examples of purposeful introduction of nonholonomy in robotic systems can be found in [19, 126, 85]. In the following, we will assume that all mechanical systems we consider are underactuated, but maximally non-holonomic, and hence controllable. Details on controllability conditions for underactuated systems can be found in most modern textbooks on nonlinear control, e.g. [98, 49, 114].

2.1.2 Mechanical control systems on Lie Groups

For many systems, in particular vehicles, the configuration manifold Q has the additional structure of a (matrix) Lie group. In such cases, it is often possible to exploit the additional

properties deriving from the Lie group structure to simplify the dynamics equations. For a concise, yet rigorous and detailed discussion of this subject, see [97].

In many cases of interest the dynamics equations are greatly simplified if written in a special reference frame, the *body frame*, which is rigidly attached to the system itself. The configuration of the system is identified by the matrix g, belonging to a matrix Lie group G. If we consider a smooth curve $g(t)$ describing the evolution of the configuration over time, its rate of change $\dot{g}(t)$ can be expressed as $\dot{g} = g\xi$ (for a left-invariant system [97]). The vector ξ represents the *velocity in body frame*. The vector ξ is an element of T_eG, the tangent space to G at the identity $e \in G$; in this case g assumes the role of a tangent map, relating vector fields on T_eG to vector fields on T_gG. The tangent space at the identity T_eG can be given the structure of a Lie Algebra, which will be denoted by $\mathfrak{g}$.

A Lie algebra is a vector space endowed with a skew-symmetric bilinear operation, the Lie bracket $[\cdot,\cdot]$, satisfying the Jacobi identity $[[\xi,\eta],\zeta] + [[\zeta,\xi],\eta] + [[\eta,\zeta],\xi] = 0$. We will also use the notation $\mathrm{ad}_\xi \eta := [\xi,\eta]$. Let $\mathfrak{g}^*$ denote the dual space of $\mathfrak{g}$, and ad^*_ξ the dual operator of ad_ξ, defined by $\langle \omega, \mathrm{ad}_\xi \eta\rangle = \langle \mathrm{ad}^*_\xi \omega, \eta\rangle$ for all $\omega \in \mathfrak{g}^*$, $\xi,\eta \in \mathfrak{g}$. We will also make use of the operator $\mathrm{Ad}_g \xi : \mathfrak{g} \to \mathfrak{g}$ which transforms velocity vectors from the body reference frame (in the configuration g) to the inertial reference frame. In other words, if $\eta = \mathrm{Ad}_g \xi$, then we can write $\dot{g} = g\xi = \eta g$.

In a matrix Lie group, the group operation is matrix multiplication. The corresponding Lie algebra is also a matrix Lie algebra, with the Lie bracket operation given by matrix commutation: $\mathrm{ad}_\xi \eta = [\xi,\eta] = \xi\eta - \eta\xi$. We also have that $\mathrm{Ad}_g \xi = g\xi g^{-1}$. The pairing $\langle p, \xi\rangle$ can be written as $\mathrm{Tr}(p^T\xi)$. We can define the exponential map $\exp : \mathfrak{g} \to G$ as the matrix exponential of elements of the Lie algebra. The local inverse of the exponential map is the logarithmic map; such a map provides a local chart on G, near the identity.

Again, following [22], we define:

Definition 2.2 (Mechanical control system on a Lie group) *A mechanical control system $\mathcal{S}$ on a matrix Lie group is described by the following objects:*

1. *An n-dimensional matrix Lie group G, defining the configuration manifold.*
2. *An inertia tensor $\mathbb{I} : \mathfrak{g} \to \mathfrak{g}^*$ on the Lie algebra $\mathfrak{g}$, defining the kinetic energy.*
3. *A potential energy $V : G \to \mathbb{R}$.*
4. *A map $b : G \times \mathfrak{g} \to \mathfrak{g}^*$, describing the action of non-conservative forces, in a body-fixed frame.*
5. *A set of covectors $\mathbf{f} = \{f^1, \ldots, f^{n_u+n_w}\} \subset \mathfrak{g}^*$, defining the input force directions, in a body-fixed frame.*
6. *A compact input set $\mathcal{U} \times \mathcal{W} \subset \mathbb{R}^{n_u} \times \mathbb{R}^{n_w}$.*

Note that in the above the inertia tensor does not depend on the configuration variables.

Given the scalar function V defined on G, and a smooth time-parameterized curve $\gamma : t \mapsto g(t)$, we indicate by $\mathrm{dV} \in T^*G$ the covector such that the time derivative of V along γ is given by $\langle \mathrm{dV}, \dot{g}\rangle$. We can define the dual g^* to the tangent map g (here we think of g as an operator from T_eG to T_gG) in the natural way, that is $\langle g^*\mathrm{dV}, \xi\rangle = \langle \mathrm{dV}, g\xi\rangle$. It is possible to show (see [22] for a derivation) that Eq. (2.2) translates into the following equations:

$$\dot{g} = g\xi \tag{2.3}$$

$$\dot{\xi} = \Xi(g,\xi,u) = \mathbb{I}^{-1}\left(\mathrm{ad}^*_\xi \mathbb{I}\xi - g^*\mathrm{d}V + b(g,\xi) + \sum_{i=1}^{m} f_i u_i\right). \tag{2.4}$$

The first equation above describes the kinematics of the system, whereas the second equation (also known as Euler-Poincaré equation) describes its dynamics.

To simplify notation, we can define the *state* of the system as the couple $x = (g,\xi)$. Such state belongs to a $2n$-dimensional manifold $\mathcal{X} = G \times \mathfrak{g}$, where n is the number of degrees of freedom of the system. We can also rewrite the dynamics equations in the compact form $\dot{x} = f(x,u)$.

2.2 Symmetry

A fundamental property of many mechanical systems, and arguably one of the defining properties of vehicles, is given by the existence of symmetries in the system's dynamics.

Roughly speaking, a symmetry is a group action on the state under which the dynamics are invariant. More precisely, consider a group $H \subseteq G$ and define its left action Ψ on the state $x = (g,\xi)$ in the following way: $\Psi : H \times \mathcal{X} \to \mathcal{X}$, $\Psi(h,(g,\xi)) = (hg,\xi)$. In the following, we will use the notation $\Psi_h(\cdot)$ as a shorthand for $\Psi(h,\cdot)$. If we assign an input history $u : \mathbb{R} \to \mathcal{U}$, we can integrate equations (2.4) from the initial conditions x_0, to obtain the *state flow*, that is a mapping $\phi_u : \mathbb{R} \times \mathcal{X} \to \mathcal{X}$, $x(t) = \phi_u(t,x_0)$ which describes the evolution of the state under the assigned input. We say that H is a symmetry group if the state flow $\phi_u(t,\cdot)$ and the group action Ψ_h commute*:

Definition 2.3 (Symmetry group) *Consider a mechanical control system on a Lie group. A group $H \subseteq G$ is said a symmetry group if*

$$\Psi_h \circ \phi_u(t,x_0) = \phi_u(t,\Psi_h(x_0))$$

for all $h \in H$, $x_0 \in \mathcal{X}$, $t \in \mathbb{R}$, and for all control signals $u : \mathbb{R} \to \mathcal{U}$.

The following condition is helpful in finding symmetry groups for a mechanical control system:

Proposition 2.1 (Condition for symmetries) *Given a mechanical control system on a Lie group, a group $H \subseteq G$ is a symmetry group if and only if the Euler-Poincaré equation (2.4) is invariant with respect to the group action Ψ_h, that is $\Xi(g,\xi,u) = \Xi(hg,\xi,u)$ for all $h \in H$, $g \in G$, $\xi \in \mathfrak{g}$,and $u \in \mathcal{U}$.*

A special case of the above condition consists of the invariance of the Lagrangian with respect to the symmetry group action in the case of conservative, autonomous systems. In coordinates, if the Lagrangian L does not depend on one of the coordinates q_i (which is then said *cyclic*, or *ignorable*), then the corresponding generalized momentum $p_i := \frac{\partial L}{\partial \dot{q}_i}$ is conserved, that is p_i is an integral of motion. This leads to a simplification of the effective

*An additional symmetry group for time-invariant systems is translation in time; a system is said time invariant if translation in time commutes with the state flow

dynamics, since the dimension of the overall system can be reduced by two. Reduction by symmetry is an extremely powerful technique to simplify the analysis of mechanical systems [4, 87].

If control inputs are available, it can be possible to artificially make one or more of the coordinates cyclic. In other words, it can be possible to modify the overall force field in such a way that it becomes integrable (i.e. it looks like it is derived from a potential), and no longer depends on one or more of the coordinates (see also the method of controlled Lagrangians [14]). If this is the case, one would expect a substantial simplification in the control problem. Nonetheless, it has been argued [113] that symmetries have not been used as much in control theory as they have in physics and mechanics (with some notable exceptions, e.g. [13, 22, 24]).

In Chapter 1, we introduced a class of internal constraints (1.1), which share the symmetry properties of the system's dynamics. By this we mean that the functions describing the internal constraints are invariant with respect to group action on the symmetry group H:

$$F(x, u) = F(\Psi_h(x), u)$$

for all $h \in H$.

In this dissertation, we will make extensive use of the concept and the properties of symmetries to identify "natural" trajectories of mechanical control systems (in the following and in Chapter 3), and to define and design invariant feedback control laws (in Chapters 4 and 5).

2.2.1 Relative equilibria

Assume that a mechanical control system admits a symmetry group H, and consider the associated Lie algebra $\mathfrak{h}$. It might be possible to set the initial conditions $x_0 = (g_0, \xi_0)$ and find a constant control input u_0 in such a way that:

- the velocity expressed in the body frame is constant: $\Xi(g_0, \xi_0, u_0) = 0$.
- the velocity expressed in the inertial frame is an element of the Lie algebra of H (and hence its exponential is an element of H): $\mathrm{Ad}_{g_0}(\xi_0) = \eta_0 \in \mathfrak{h}$.

In such a case, it is easy to verify (remember that for a matrix Lie group $Ad_g\xi = g\xi g^{-1}$, and the definition of symmetry group) that the evolution of the system will be given by:

$$\begin{aligned} g(t) &= \exp(\eta_0 t) g_0, \\ \xi(t) &= \xi_0. \end{aligned}$$

We call such a trajectory a *relative equilibrium* of the system. Along these trajectories the velocities in body axes ξ and the control inputs u are constant: in the aerospace literature such relative equilibria are also known as *trim trajectories.*

Notice that for simple Lagrangian systems, existence of relative trajectories follows from Noether's theorem in the presence of cyclic coordinates. For non-conservative systems, there is no such direct relationship between symmetries and relative equilibria. However, it is possible to argue that all human-built vehicles are built in such a way that relative equilibria do indeed exist. If this were not the case, pilots should always be adjusting the control inputs, even in nominal situations (i.e. in the absence of wind gusts, bumps in the road, etc.); this would easily result in excessive workloads.

2.3 Examples

In this section we will present a few examples of vehicles or benchmark systems for which we will address the motion planning problem in the rest of the dissertation.

2.3.1 System with integrators

As a simple example, consider a mechanical control system with no potential forces, and configuration-independent non-conservative forces and control inputs. The dynamics are described by the following equations:

$$\begin{aligned} \dot{q} &= p \\ \dot{p} &= f(p,u) \end{aligned}$$

where $q, p \in \mathbb{R}^n$. The configuration manifold is $G = \mathbb{R}^n$, and the group operation is the vector addition (G is an Abelian Lie group). The Lie algebra $\mathfrak{g}$ is $\mathbb{R}^n$ itself, with the trivial Lie bracket $[p_1, p_2] = 0$, for all $p_1, p_2 \in \mathbb{R}^n$.

It is readily seen that any group action that translates q leaves the dynamics unchanged: the symmetry group H coincides with the configuration manifold G. Relative equilibria correspond to couples $(\bar{p}, \bar{u})$ such that $f(\bar{p}, \bar{u}) = 0$, and result in trajectories of the form $q(t) = q_0 + \bar{p}t$, $p(t) = \bar{p}$ (the exponential map on $(\mathbb{R}^n, +)$ is the identity map).

2.3.2 Three degrees of freedom helicopter

Consider the three degrees of freedom (3-DOF) helicopter setup shown in Fig. 2-1. The setup consists of a helicopter body carrying two propellers, which can be commanded independently. The body of the helicopter is attached to an arm, and is free to rotate about the axis of the arm. The arm, in turn, is attached to the ground via a spherical joint.

If we make the assumption that the center of mass of the helicopter body lies on the longitudinal axis of the arm, and neglect the moment of inertia of the arm about its longitudinal axis (i.e. we consider the arm to be a long and thin rod), we can think of the whole assembly as a rigid body rotating about a pivot point located on the spherical joint. The configuration will hence be defined by a rotation matrix $R \in SO(3)$. The group $SO(3)$ is known as the *special orthogonal group* in 3 dimensions, and is the set of all 3×3 matrices with determinant equal to $+1$. The associated Lie algebra $\mathfrak{so}(3)$ is the set of skew-symmetric matrices in $\mathbb{R}^{3\times 3}$.

If we define the "hat" operator $\cdot : \mathbb{R}^3 \to \mathfrak{so}(3)$ as follows:

$$\hat{x}y := x \times y \;\; \forall x, y \in \mathbb{R}^3 \tag{2.5}$$

where $\times$ represents the vector cross product, we can identify $\mathfrak{so}(3)$ and $\mathbb{R}^3$. In other words, $\hat{x}$ represents the 3×3 matrix which has the same effect on a vector y as vector cross multiplication by x. If $x = [x_1, x_2, x_3]^T$, then

$$\hat{x} = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}.$$

The kinematics of the system are described by $\dot{R} = R\hat{\omega}$, where $\omega \in \mathbb{R}^3$ is the angular velocity in body axes.

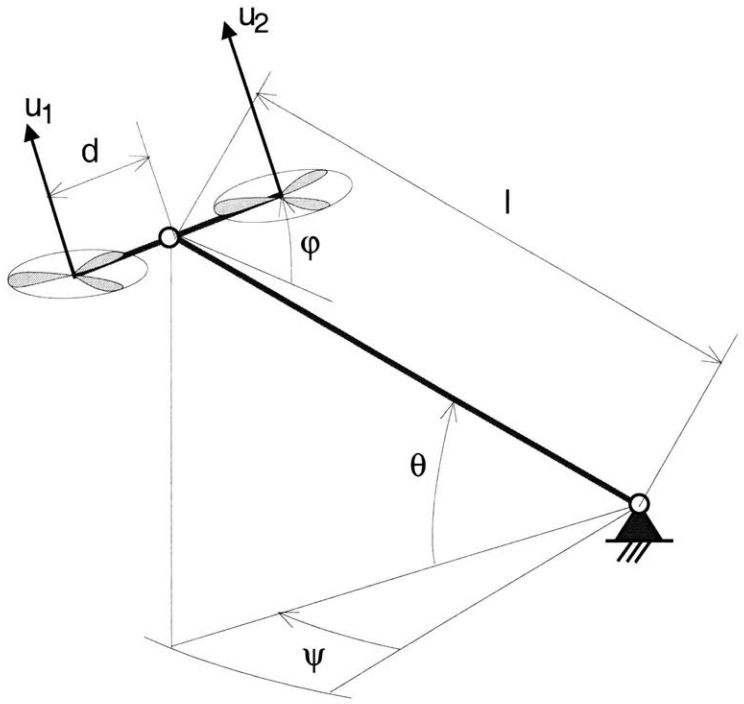

Figure 2-1: Three degrees of freedom helicopter.

In the following, we will neglect the gyroscopic effect due to the angular momentum of the propellers, and also aerodynamic drag effects (negligible at low speed). The system is subject to gravitational forces, defining the potential energy $V(R) = -mg^T R\lambda$, where m is the mass of the system, g is the vector defining gravity acceleration, and λ is the vector defining the position of the center of mass with respect to the pivot point, in body axes. The dynamics of the system are given by

$$J\dot{\omega} = \mathrm{ad}^*_\omega J\omega - R^* \mathrm{dV} + f(u),$$

that is:

$$J\dot{\omega} = -\omega \times J\omega + m\lambda \times R^T g + f(u), \tag{2.6}$$

where $J = \mathrm{diag}\{J_1, J_2, J_3\}$ represents the moments of inertia with respect to the pivot point (in a suitably chosen body frame).

The torques in body axes are given by:

$$f(u) = [d(u_2 - u_1), l(u_1 + u_2), 0]^T, \tag{2.7}$$

where d is the distance from the propellers to the main arm, and l is the moment arm of the propellers with respect to the pivot point.

As it can be recognized from Eq.(2.6) the dynamics are invariant with respect to rotations about a vertical axis ($R^{-1}g$ is the only term involving the configuration of the helicopter). The symmetry group is in this case the group $S^1 \subset SO(3)$ of rotations about a vertical axis.

Trim trajectories can be found by solving for $\dot{\omega} = 0$, and imposing the constraint that $R\omega$ has the form $\eta_0 = [0,\ 0,\ \dot{\psi}_0]^T$. In coordinates:

$$\begin{aligned}(J_2 - J_3)\omega_2\omega_3 &= d(u_1 - u_2)\\ (J_3 - J_1)\omega_1\omega_3 - mg\lambda\cos\phi\cos\theta &= l(u_1 + u_2)\\ (J_1 - J_2)\omega_1\omega_2 + mg\lambda\sin\phi\cos\theta &= 0.\end{aligned}$$

In our case, relative equilibria are given by trajectories along which

$$\begin{aligned}R(t) &= \exp(\hat{\eta}_0 t)\begin{bmatrix}\cos\theta & 0 & \sin\theta\\ 0 & 1 & 0\\ -\sin\theta & 0 & \cos\theta\end{bmatrix}\\ \omega &= [-\dot{\psi}\sin\theta, 0, \dot{\psi}\cos\theta]^T\\ u_1 = u_2 &= \frac{1}{2l}\left(mg\lambda\cos\theta + \frac{J_3 - J_1}{2}\dot{\psi}_0^2\sin 2\theta\right).\end{aligned}$$

2.3.3 Small autonomous helicopter

The small helicopter model that we will use here is based on the models presented in [62, 38]. A similar model has been widely used in the nonlinear control literature, in the three degrees of freedom case, as a VTOL aircraft model [42, 114]. More details on helicopter dynamics can be found in [51, 107, 102].

The dynamics of small model helicopter can be adequately described by the rigid body equations [97]. The configuration of the vehicle will be described by an element g of the Special Euclidean group in the three-dimensional space, denoted by $SE(3)$. Using homogeneous coordinates, a matrix representation of $g \in SE(3)$ is the following:

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

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

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

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

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

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 $u \in \mathbb{R}^3$. As mentioned in the previous section, through the use of the "hat" operator, we can identify $\mathfrak{se}(3)$ and $\mathbb{R}^6$. The dynamics equations will be given by:

$$\mathbb{I}\dot{\xi} = \mathrm{ad}^*_\xi \mathbb{I}\xi - g^*\mathrm{dV} + B(g,\xi) + f(u).$$

The adjoint operators can be written in the following way:

$$\begin{aligned}\mathrm{ad}_{(\omega,v)} &= \begin{bmatrix}\hat{\omega} & 0\\ \hat{v} & \hat{\omega}\end{bmatrix}\\ \mathrm{Ad}_{(R,p)} &= \begin{bmatrix}R & \hat{p}R\\ 0 & R\end{bmatrix}.\end{aligned}$$

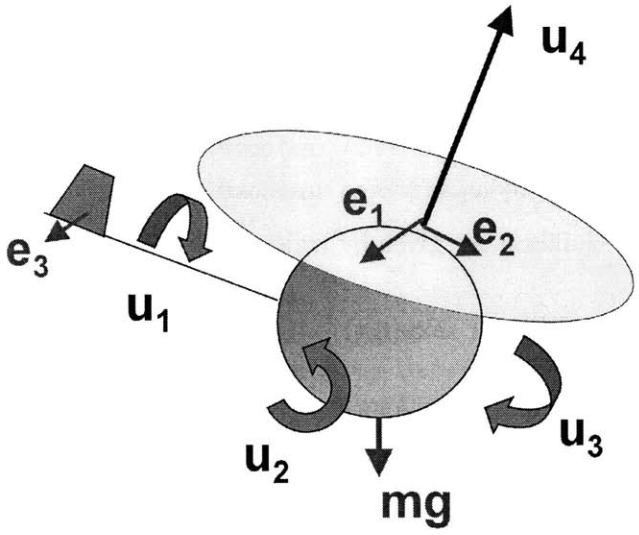

Figure 2-2: Helicopter model.

Finally, the Euler-Poincaré equations become:

$$J\dot{\omega} = -\omega \times J\omega + \beta(g,\xi) + \tau(u) \tag{2.8}$$

$$m\dot{v} = -\omega \times mv + mR^{-1}\tilde{g} + b(g,\xi) + f(u), \tag{2.9}$$

where J and m are the vehicle's rotational inertia tensor and mass, $\tilde{g}$ is the gravity acceleration, and τ, β and f, b represent the control and non-conservative torques and forces in body axes.

We can use the following form for the control force and moment in Eq. (2.8):

$$\begin{cases} f = u_4 \left[0,0,-1\right]^T + \mathcal{E}(u_1,u_2,u_3) \\ \tau = \left[u_1,u_2,u_3\right]^T \end{cases} . \tag{2.10}$$

In the above, we have defined:

$$\mathcal{E}(u_1,u_2,u_3) = \begin{bmatrix} 0 & -\epsilon_2 & 0 \\ \epsilon_1 & 0 & -\epsilon_3 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 - \epsilon_4(u_4) \end{bmatrix}$$

where:

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

and:

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

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 coefficients in the approximate expression of the main rotor reaction torque (which can be obtained, for example, through momentum theory, as in Johnson [51]). We will assume that T,a_1,b_1, and F_t are directly controlled through the collective and cyclic main rotor pitch commands, throttle setting, and tail rotor collective pitch command.

The non-conservative forces and moments include aerodynamic drag, dihedral, and attitude damping terms:

$$\beta = \begin{bmatrix} -L_p(\xi)\omega_1 - L_v(\xi)v_2 \\ -M_q(\xi)\omega_2 + M_u(\xi)v_1 \\ -N_r(\xi)\omega_3 \end{bmatrix}$$

$$b = \begin{bmatrix} -X_u(\xi)v_1 \\ -Y_v(\xi)v_2 \\ -Z_w(\xi)v_3 \end{bmatrix}.$$

Again, we notice that in Eq.(2.8) the only term that depends on the configuration (R, p) is the term containing the gravitational force, $mR^{-1}\tilde{g}$. As a consequence, the dynamics of the helicopter are invariant with respect to translations, and also with respect to rotations about a vertical axis. The symmetry group in this case can be identified with the four-dimensional group $\mathbb{R}^3 \times S^1$ (or, equivalently, with the group $SE(2) \times \mathbb{R}$ of rigid body motions on the plane composed with vertical translations). All trim trajectories will be the composition of a screw motion $h(t) \in H$, given by the exponential of an element $\hat{\eta}$ of the Lie sub-algebra $\mathfrak{h} \subset \mathfrak{se}(3)$, and a constant rigid body motion g_0. The screw motion $h(t)$ 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 parameters $\left\{V, \gamma, \dot{\psi}, \beta\right\}$, where V is the magnitude of the velocity vector, γ is the flight path angle (describing the angle at which the vehicle is climbing or descending), $\dot{\psi}$ is the turning rate, and finally β is the sideslip angle, measuring the amount of "skidding" of the vehicle, i.e. the ratio of lateral vs. longitudinal motion.

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

$$h = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & x \\ \sin\psi & \cos\psi & 0 & y \\ 0 & 0 & 1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.11}$$

Elements of the Lie algebra $\mathfrak{h}$ can be represented as:

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

Finally, we can derive the general form of all relative equilibria:

$$\begin{aligned} g(t) &= \exp(\hat{\eta}t)g_0 = \\ &= \begin{bmatrix} \cos\psi(t) & -\sin\psi(t) & 0 & \rho(\sin\psi(t)\cos\beta + \cos\psi(t)\sin\beta - \sin\beta) \\ \sin\psi(t) & \cos\psi(t) & 0 & \rho(\sin\psi(t)\sin\beta - \cos\psi(t)\cos\beta + \cos\beta) \\ 0 & 0 & 1 & Vt\sin\gamma \\ 0 & 0 & 0 & 1 \end{bmatrix} g_0 \end{aligned} \tag{2.12}$$

where $\psi(t) = \dot{\psi}t$ (and $\dot{\psi}$ is a constant), and we have introduced the signed turning radius $\rho := (V\cos\gamma)/\dot{\psi}$.

In the case in which $\mathcal{E} = 0$, and in the absence of non-conservative forces, it is relatively easy to find relative equilibria. We have to set $\dot{\xi}$ to zero, and make sure that $\mathrm{Ad}_{g(t)}\,\xi(t) = \mathrm{Ad}_{g(t)}\,\xi_0 \in \mathfrak{h}$. From Eq. (2.8), and working in inertial coordinates, we see that the control forces $u_4 R[0,\ 0\ , -1]^T$ must balance the weight of the helicotper $m\tilde{g}$, and the centrifugal terms $-\omega_I \times m v_I$ (the subscript I indicates objects expressed in inertial coordinates).

Assuming, for simplicity, that the desired sideslip angle is zero, we get the following equation for the balance of forces:

$$mV \begin{bmatrix} 0 \\ -\dot{\psi}\cos\gamma \\ 0 \end{bmatrix} + m\tilde{g} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} + u_4 \begin{bmatrix} 0 \\ \sin\phi \\ -\cos\phi \end{bmatrix} = 0,$$

which reduces to:

$$\tan\phi = \frac{V\dot{\psi}\cos\gamma}{\tilde{g}} = \frac{\rho\dot{\psi}^2}{\tilde{g}}$$

$$u_4 = \frac{m\tilde{g}}{\cos\phi}.$$

Notice that, because of the symmetry of the system, we have set x,y,z,and ψ to zero, without loss of generality.

The angular velocity in body axes is given by $\omega = \left[0, -\dot{\psi}\cos\phi, -\dot{\psi}\sin\phi\right]^T$, and the values of the control inputs u_1,u_2, and u_3 are found by solving the moment balance equation:

$$-\omega \times J\omega + \beta(\xi) + \tau(\xi, u) = 0.$$

2.3.4 General vehicle models

The symmetry group that we just derived for the helicopter, along with the relative equilibria, characterizes the dynamics of a very large class of human-built vehicles. Examples include aircraft, ground vehicles such as unicycles, bicycles, cars, trailers, surface and underwater vessels such as motor boats, hovercraft, and submarines. The basic assumption in all cases is that the motion occurs through a homogeneous and isotropic medium (i.e. constant density, no wind), and that gravity is a constant vector field. Notable exceptions are:

- An airplane undergoing large altitude changes: In this case, since the density of the atmosphere, and hence the handling characteristics of the airplane, change with altitude, the symmetry group is $SE(2)$, the group of rigid body motion at constant altitude. The resulting relative equilibria consist of straight lines and circular arcs on a horizontal plane.

- A sailboat: Since the dynamics of a sailboat obviously depend on the wind, and on the relative position of the sail with respect to it, the isotropic atmosphere assumption can not be made. The direction of the wind breaks the rotational symmetry of the system, and the symmetry group is the group of translations $\mathbb{R}^2$ (the motion of the sailboat is restricted to the surface of the sea). Relative equilibria are straight lines, corresponding to the different sailing points, such as broad or beam reach, close hauled and downwind sailing.

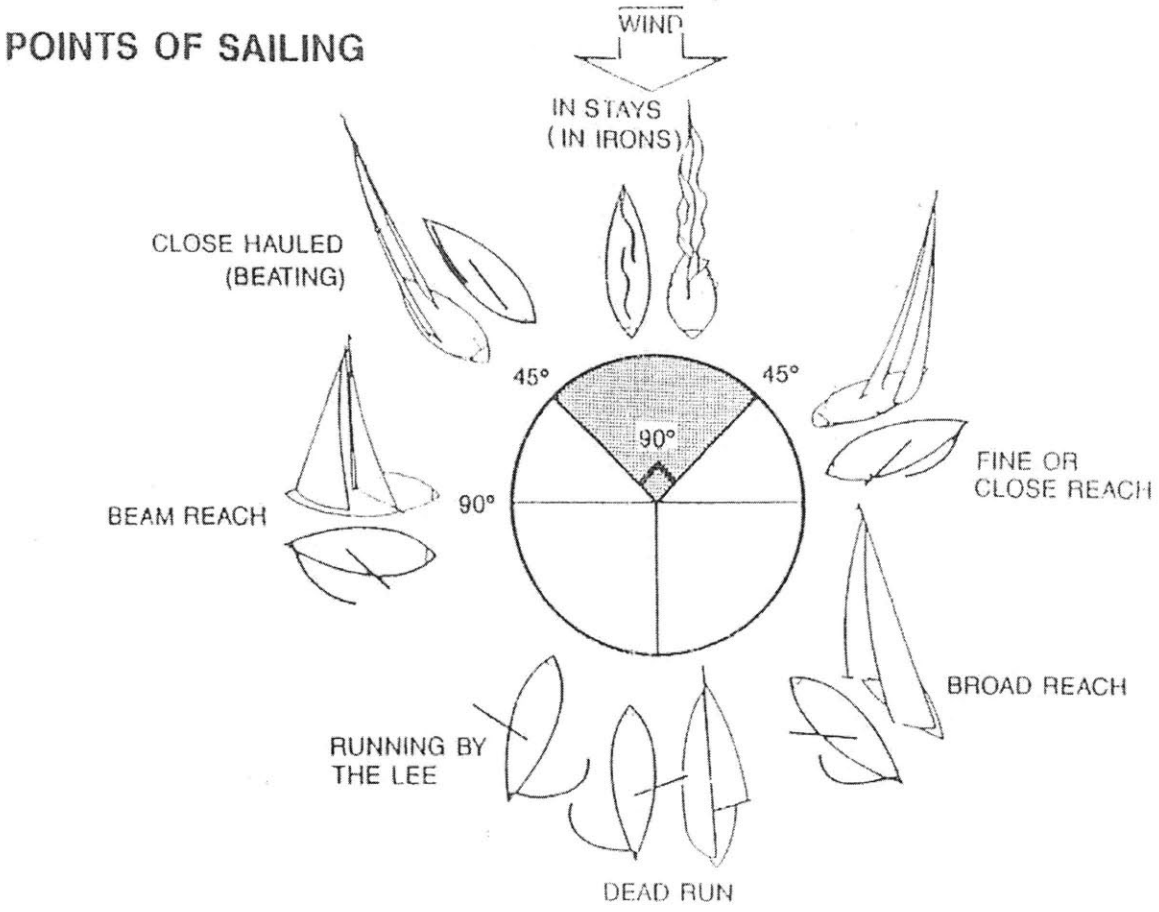

Figure 2-3: Relative equilibria for a sailboat.

Chapter 3

The Maneuver Automaton

In this chapter we address the basic problem of motion planning for an autonomous vehicle, described as a nonlinear mechanical control system, in a free environment. For the purpose of the framework presented in this chapter, we assume that the dynamics of the systems are completely known, and there are no external disturbances on the system. Hence we concentrate, for the time being, with the nominal dynamics of the system; extensions for robustness to uncertainties and disturbances will be discussed in the next chapter.

In this chapter, a new control framework for planning the motion of underactuated mechanical control systems is introduced, which builds upon the interconnection of appropriately selected trajectory primitives. First we define the concept of trajectory primitive, and define two classes of trajectory primitives that are of particular interest, and which will form the basis of our motion planning framework . Then we define the maneuver automaton, that is a dynamical system which outputs all possible valid interconnections of trajectory primitives. The maneuver automaton represents a new dynamical system, whose properties will be analyzed. Finally, we give details on the solution of optimal control problems on the maneuver automaton, and relate such solutions to the solutions obtained from the full, continuous system.

3.1 Maneuver automaton

The approach that we propose to reduce the computational complexity of the motion planning problem for a constrained, nonlinear, high dimensional system, is based on a quantization of the system dynamics. By quantization of the system dynamics we mean that we restrict the feasible nominal system trajectories to the family of time-parameterized curves that can be obtained by the interconnection of appropriately defined primitives. In this sense, we are not introducing a state, control, or time discretization, but rather we want to reduce the complexity of motion planning problems (including steering and optimal control in a free or cluttered environment) by quantizing the possible "choices" to be made at each decision step. Naturally, such quantization must preserve the desirable characteristics of the system's dynamics, and possibly to approximate the "quality" of the trajectories that can be achieved through the solution to the full optimal control problem.

As a consequence, the main task in developing such a framework is to define and select its building blocks, that is, the trajectory primitives which will eventually build the maneuver library.

3.1.1 Trajectory primitives

The class of systems to which our approach applies is characterized by the existence of symmetries and relative equilibria, as defined in Chapter 2. This class is very large, and includes many systems of practical interest. As a matter of fact, the existence of symmetries and relative equilibria can arguably be seen as one of the defining factors for the general class of mechanical systems which go under the name of "vehicles". We can state the first main assumption under which our framework is applicable:

Assumption 3.1 (Existence of symmetries) *The mechanical control system $\mathcal{S}$ admits a (non-trivial) symmetry group $H \subseteq G$.*

As a matter of fact, the dynamics of most human-built vehicles, including bicycles, cars, aircraft, ships, etc., under fairly reasonable assumptions (such as homogeneous and isotropic atmosphere, constant gravity acceleration) are invariant with respect to translations, and rotation about a vertical axis (i.e. an axis parallel to the direction of the local gravitational field). This has a simple "engineering" explanation: vehicles are designed such that once one has learned how to operate them at some location (e.g. in the backyard, at the driving school, etc.), he/she can apply the same skills to operate the vehicle anywhere in the world. In such a case the symmetry group H corresponds to $H = SE(2) \times \mathbb{R} \equiv \mathbb{R}^3 \times S^1 \subset SE(3)$. An element $h \in H$ is completely described by a translation vector $p \in \mathbb{R}^3$ and a heading angle $\psi \in S^1$.

The property of symmetry induces an equivalence relation between trajectories:

Definition 3.1 (Trajectory equivalence) *Consider two trajectories, $g_1, g_2 : [t_0, t_f] \mapsto G$. The trajectories g_1 and g_2 are said to be equivalent on the interval $[t_0, t_f]$ if there exists an element $h \in H$ of the symmetry group such that $g_2(t) = hg_1(t)$ for all $t \in [t_0, t_f]$.*

Two trajectories which are equivalent under the above definition can be transformed exactly to one another by the (left)-action of an element of the symmetry group. Hence we can formalize the notion of a trajectory primitive:

Definition 3.2 (Trajectory primitive) *We define a trajectory primitive as an equivalence class of trajectories, under the trajectory equivalence relation.*

Since trajectory primitives are an equivalence class of trajectories, without loss of generality each equivalence class can be represented by a prototype, starting at the identity on the symmetry group. The notion of symmetry thus becomes fundamental in the construction of a "library" of trajectories.

At this point we have to address the problem of selecting an appropriate set of trajectory primitives from among all possible trajectory (modulo a symmetry transformation) which can be executed by the mechanical control system at hand.

We want to characterize trajectory primitives in order to

- capture the relevant characteristics of the system's dynamics,
- allow the creation of complex behaviors from the interconnection of primitives (we want to obtain "good" approximations to optimal solutions),
- 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.

3.1.2 Relative equilibria

A first class of trajectory primitives is composed of relative equilibria, or trim trajectories, as defined in Section 2.2.1. To fix ideas, relative equilibria are steady-state trajectories of the system, in which the velocities in body axes (i.e. as perceived by a "pilot") and the control input are constant. Relative equilibria are strongly connected to the existence of symmetries in the system's dynamics. In fact, these correspond to the existence of conserved quantities and integral of motions for Lagrangian systems, through Noether's theorem [4, 87]. In the following, we will make the following fundamental assumption:

Assumption 3.2 (Existence of relative equilibria) *The mechanical control system $\mathcal{S}$ admits relative equilibria.*

As in the case of symmetries, most human-built vehicles do indeed satisfy Assumption 3.2. Again, there is a simple "engineering" reason behind this. A relative equilibrium corresponds to a trajectory in which the pilot/driver/helmsman can hold the control inputs fixed, at least in the nominal case. The workload of the pilot in this case is very small (assuming good handling qualities), and restricted to small corrections to compensate for the effect of the environmental disturbances, such as wind gusts, bumps on the road, waves, etc. In the absence of relative equilibria, the pilot would be required to perform continuous corrections to the controls, even in the nominal case, probably resulting in a very high workload.

Consider a mechanical control system on a Lie group (defined in Section 2.1.2). Its configuration is determined by $g \in G$, and its velocity in body axes is determined by $\xi \in \mathfrak{g}$. The kinematics are described by $\dot{g} = g\xi$, while the dynamics are described by the Euler-Poincaré equations (2.4), that we write in the form

$$\dot{\xi} = \Xi(g, \xi, u).$$

As detailed in Section 2.2.1, if $H \subseteq G$ is a symmetry group for the system, then trim trajectories are defined by the following:

- A constant control setting $\bar{u}$,
- a constant configuration $\bar{g}$,
- a constant vector $\bar{\xi}$, with $\bar{\eta} := \mathrm{Ad}_{\bar{g}}\bar{\xi} \in \mathfrak{h}$,

with the condition that $\Xi(\bar{g}, \bar{\xi}, \bar{u}) = 0$. As a consequence, any instance of a given trim trajectory, can be expressed as:

$$g(t) = h(0)\exp(\bar{\eta}t)\bar{g}, \tag{3.1}$$

where $h(0)$ indicates the initial displacement on the symmetry group (with respect to the initial conditions of the prototype). From the definition of relative equilibria, it is apparent that relative equilibria are characterized by an infinite time horizon: Once a trim trajectory is initiated, it can be followed indefinitely (it is a steady state condition).

Note that relative equilibria include trivially all equilibrium points of the system (e.g. points for which $\dot{g} = 0$, $\dot{\xi} = 0$). The simplest possible motion primitive is then represented by equilibrium points. In a system with multiple equilibrium points, each equilibrium point can be chosen as a trajectory primitive. In the case in which the system admits only a

trivial relative equilibrium (i.e. a single equilibrium point), the framework we are going to present in this chapter reduces to a version of the control quanta methodology (see Section 1.1.2).

In the general case of an unconstrained mechanical control system, the collection of all possible trim trajectories defines, under some smoothness conditions on the system dynamics, an m-dimensional manifold in $\mathcal{X} \times \mathcal{U}$, known as the *trim surface* (where m is the number of independent control inputs). When the flight envelope and control saturation constraints are taken into account, the relative equilibria that are actually attainable by the system are typically contained in a compact set.

It is important to remark that, given a desired velocity in body axes $\bar{\eta}$, there might exists several combinations of $\bar{g}$ and $\bar{u}$ which result in a trajectory described by Eq. (3.1). It is a matter of design to choose the most desirable ones. As an example, a model airplane, or a model helicopter, can usually be trimmed in forward flight at a given velocity either upside-down or right side up.

Another interesting fact is that the set of feasible relative equilibria is not necessarily connected, as in the case of a sail boat (with a triangular sail), which can only be trimmed either on a starboard tack or a port tack (i.e. with the main sail either on the "left" or on the "right" of the longitudinal axis of the boat, see Figure 2-3). Changing from a starboard to a port tack is impossible while staying always close to equilibrium, and requires highly unsteady maneuvers:

- **Tacking:** A tacking maneuver is required when it is desired to change tack moving through the upwind direction. Since it is clearly impossible to sail directly upwind, this transition must be performed with a quick transient.

- **Jibing:** A jibe is required when it is desired to change tack moving through the downwind direction. When sailing downwind, the sail is almost completely deployed to one side of the boat (if safe and efficient sailing is desired). To change tack, the sail must be moved by a large angle to the other side of the boat.

Trim trajectories are at the foundation of the vast majority of control design methods. Linear control is typically based on the Jacobian linearization of a nonlinear system's dynamics around equilibria, or relative equilibria (steady-state conditions). Moreover, the class of trim trajectories 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 [73, 133, 92, 26]. The ideas of gain scheduling and of Linear Parameter Varying (LPV) system control can also be brought into this class [119], as well as other integrated guidance and control systems for UAV applications [53].

However, restricting the choice of trajectory primitives to trim trajectories alone has several drawbacks. First of all there is a lack of hierarchical consistency. A dynamical system cannot transition instantaneously from one trim trajectory to another one, since there will always be a transient between the two. Such a selection of trajectory primitives is appropriate for systems in which the dynamic behavior can be neglected (e.g. kinematic systems, and aircraft models for Air Traffic Control). If the dynamics of the system cannot be neglected, and aggressive maneuvering is desired while maintaining stability guarantees, the transients related to switches between different trim trajectories must be accounted for. Especially in the case in which the selection of the reference trim trajectory is carried out in real time in a closed-loop fashion, such as in the case of autonomous trajectory generation,

the absence of any information on the transient behavior can lead to undesirable effects, such as collision with obstacles, the onset of limit cycles, or possibly instability of the closed-loop system.

A second reason is related to the performance attainable by trim trajectories alone. Switching between trim trajectories generally results in relatively poor performance, and in "slow" transitions, as the system is required to stay in some sense close to the trim surface, where by definition some of the derivative of the state are zero.

A last reason for which trim trajectories alone could not be an appropriate set of trajectory primitives for some system is due to the fact that in some cases there could be several, disconnected regions on the trim surface. For example, it is not possible to steer a sailboat while staying always close to some equilibrium point. Transitions from a starboard tack to a port tack and vice-versa are very far from steady-state conditions.

3.1.3 Maneuvers

To address the above mentioned issues and, in general, to allow the system to perform 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 is general enough to include most of the trajectories which are executable by the system. Moreover, it is the first formal definition leading to the construction of a framework that is amenable to mathematical programming, and will eventually result in a significant reduction of the complexity of the motion planning problem. The key concept here is that the formal definition of maneuver needs to address the following points:

- When do we say that a maneuver *starts*?
- When do we say that a maneuver *ends*?
- How can we build a consistent *interface* between maneuvers, so that we can sequence them efficiently?

Our answers are summarized in the following definition:

Definition 3.3 (Maneuver) *A maneuver is defined as a finite time transition between two trim trajectories.*

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 maneuver is a transition from and back to straight forward motion). The execution of a maneuver can also connect two trim trajectories belonging to disconnected components of the feasible trim set: this is what is customarily accomplished by sailors when tacking or jibing. Finally, the requirement that maneuvers start and end at trim trajectories does not mean that the system must remain at equilibria: it must go *through* equilibria when switching from one maneuver to another: this is required to provide a set of interfaces between maneuvers (i.e. a "standard" set of connections).

Each maneuver is characterized by the following:

- A time duration T.

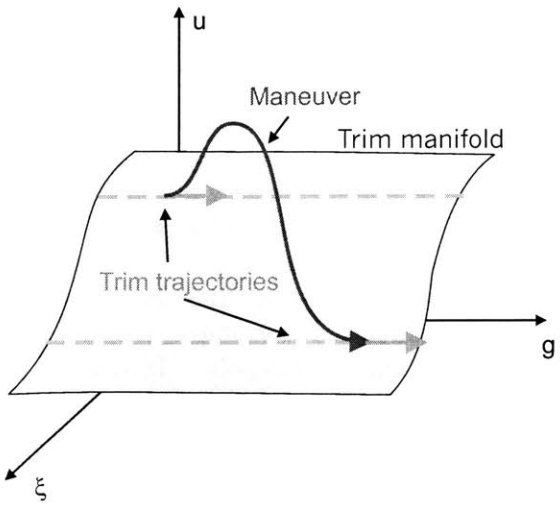

Figure 3-1: Trajectory primitives.

- A control signal $u : [0, T] \to \mathcal{U}$.
- A function $\phi [0, T] \to G \times \mathfrak{g}$, describing the evolution of the system over the maneuver.

Since a maneuver has to start and end at trim trajectories, the following boundary conditions must be satisfied:

$$\begin{array}{rclcrcl} u(0) & = & \bar{u}_{\text{start}}, & & u(T) & = & \bar{u}_{\text{end}} \\ \phi(0) & = & \left(h(0)\bar{g}_{\text{start}}, \bar{\xi}_{\text{start}}\right), & & \phi(T) & = & \left(h(T)\bar{g}_{\text{end}}, \bar{\xi}_{\text{end}}\right) \end{array}$$

for some $h(0)$, $h(T) \in H$, and with the subscripts start and end referring to the trim trajectories connected by the maneuver. Because of the symmetry of the dynamics, the displacement on the symmetry group, $h_m := h(0)^{-1}h(T)$ is invariant, and is a constant characterizing the maneuver. Hence, one of the main properties of maneuvers is the fact that each maneuver is characterized by a well-defined time duration, and results into a well-defined displacement (see Fig. 3-2). Even though the displacement on the symmetry group h_m can be derived from the function ϕ, it is convenient to store this displacement along with the other data on the maneuver.

We will not discuss here the details of how to generate the nominal state and control trajectories describing maneuvers (some examples will be given later in this chapter): 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 [105] or stabilizing controllers (as in 3.4.2), off-line solutions to optimal control problems [21, 6], or real-time trajectory generation methods [137].

Finally, we would like to remark that the design of the nominal trajectories, along with the tracking control introduced in the next section, can be carried out so as to ensure that the vehicle does not violate the "internal" constraints (1.1), that is flight envelope and control saturation constraints [122, 123, 135]. This is possible if the flight envelope constraints share the symmetry properties of the system's dynamics, that is if

$$F(x, u) = F(\Psi_h(x), u)$$

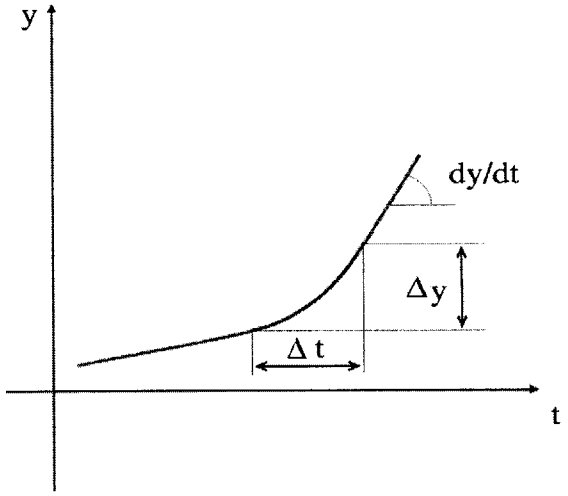

Figure 3-2: Maneuver for a 1-DOF system, transitioning between two velocity levels.

for all $h \in H$. Thus in this sense the objective of envelope protection is ensured implicitly by the construction of the maneuvers.

3.1.4 Maneuver Automaton definition

To discuss the details of the control architecture, recall that we have identified two classes of trajectory primitives, namely:

- Relative equilibria (or trim trajectories).
- Maneuvers.

The control framework we propose is based on the selection of a finite number of relative equilibria, connected by a finite number of maneuvers: such trajectory primitives will form a *maneuver library.*

It should be clear by now that the control architecture will involve switching from one trajectory primitive to another, always alternating relative equilibria and maneuvers. Such a control system will include both continuous and discrete dynamics, thus belonging to the realm of hybrid control. The continuous dynamics correspond to the dynamics of the system along a trajectory primitive. The discrete dynamics correspond to the switching between trajectory primitives, and hence to the logical state of the controller.

The *discrete dynamics* of such a system are well represented by a finite state automaton, and can be conveniently depicted by a directed graph, as in Fig. 3-3. The nodes of the directed graph correspond to trim trajectories in the automaton, whereas the edges correspond to maneuvers. The directed graph representation is a convenient form of depicting the discrete dynamics of the system, that is, all the possible sequences of maneuvers/trim trajectories that the system is able to perform. However, a simple directed graph is not enough to convey all the information contained in the automaton, since it does not show the continuous evolution: The nodes in Fig. 3-3 are represented as sections of tubes, or cylinders, to remind the reader of the fact that while the discrete state is stationary in one node, the continuous dynamics keep evolving (on a relative equilibrium).

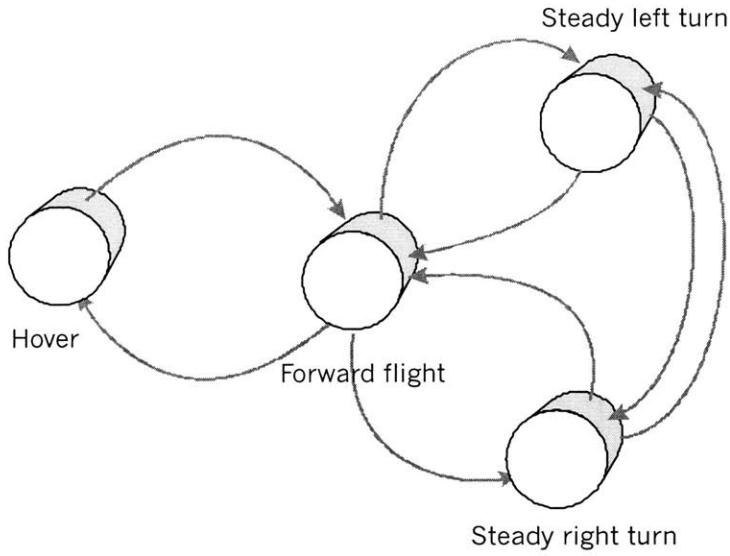

Figure 3-3: Maneuver Automaton (simplified).

From the properties and definitions of trim trajectories and maneuvers, it follows that the discrete state can stay indefinitely in any node, but it must traverse the edges in a very definite time. Also, once the discrete state switches to an edge $q \in Q_M$ (i.e. the execution of a maneuver is started), the only possible outcome is for the system to switch to the next trim trajectory, after T_q seconds.

We will call such a control architecture a Maneuver Automaton:

Definition 3.4 (Maneuver Automaton) *A Maneuver Automaton MA over a mechanical control system $\mathcal{S}$, with symmetry group H, is described by the following objects:*

- *A finite set of indices $Q = Q_T \cup Q_M \subset \mathbb{N}$, where the subscript T relates to trim trajectories, and the subscript M relates to maneuvers;*
- *A finite set of trim trajectory parameters $(\bar{g}, \bar{\xi}, \bar{u})_q$, with $q \in Q_T$.*
- *A finite set of maneuver parameters, and state and control trajectories $(T, u, \phi)_q$, with $q \in Q_M$.*
- *The maps* $\mathrm{Pre} : Q_M \to Q_T$, *and* $\mathrm{Next} : Q_M \to Q_T$ *such that* $\mathrm{Pre}(q)$ *and* $\mathrm{Next}(q)$ *give, respectively, the index of the trim trajectories from which the maneuver q starts and ends.*
- *A discrete state $q \in Q$.*
- *A continuous state, denoting the position on the symmetry group, $h \in H$.*
- *A clock state $\theta \in \mathbb{R}$, which evolves according to $\dot{\theta} = 1$, and which is reset after each switch on q.*

It is apparent that decisions can be made about the future evolution of the system only when the system is executing a trim trajectory (that is, the discrete state is in one of the

nodes in the graph). While executing a maneuver, the system is committed to it, and must keep executing the maneuver until its completion. As a consequence, for motion planning and control design purposes, we can concentrate the study of the evolution of the system on and between nodes.

While on a trim trajectory, the state of the system is *completely* determined by the couple $(q, h) \in Q_T \times H$: The corresponding full continuous state will in fact be given by $(h\bar{g}_q, \bar{\xi}_q)$. We will call the couple (q, h) the *hybrid state* of the system.

While on a node, the discrete state stays constant, and the continuous evolution is given by $g(t) = h(t)\bar{g}_q$, that is :

$$\begin{aligned} \dot{q} &= 0 \\ \dot{h} &= h\bar{\eta}_q \\ \dot{\theta} &= 1. \end{aligned}$$

At any time while in trim trajectory q, it is possible to start a maneuver p, as long as $\mathrm{Pre}(p) = q$. Once a maneuver is initiated, and executed to its completion, the hybrid state will undergo a net change, described by the following jump relations:

$$\begin{aligned} q^+ &= \mathrm{Next}(p) \\ h^+ &= h^- h_{m,p} \\ \theta^+ &= 0, \end{aligned}$$

with $h_{m,p}$ being the displacement on the symmetry group caused by maneuver p. For completeness, we can also give the complete characterization of the full continuous state during the maneuver p, initiated at time t_0 as $x(t) = \Psi_{h(t_0)}(\phi_p(t - t_0))$ (assuming that $t - t_0 < T_p$).

From the above discussion, it should be apparent that the decision variables, when executing a trim trajectory $q \in Q_T$, are the following:

- The duration of the permanence of the system in the current trim trajectory. We call this *coasting time*, and indicate it with τ.
- The next trajectory to be executed. We indicate the next trajectory as $p \in \mathrm{Pre}^{-1}(q)^*$.

At this point, along with the hybrid state (q, h), we have a hybrid control variable $\upsilon := (p, \tau)$. The action of the hybrid control on the hybrid state (e.g. the evolution of the system between two arrivals at nodes) is given by

$$(q, h) \xrightarrow{(p,\tau)} (\mathrm{Next}(p), h \exp(\bar{\eta}_q \tau) h_m) .$$

In the following, we will indicate the action of the hybrid control with the maneuver-end map $\Phi : Q_T \times H \times Q_M \times \mathbb{R} \to Q_T \times H$. Moreover, we will use the notation $\Phi_\upsilon(q, h)$ as a shorthand for $\Phi(q, h, \upsilon)$.

3.2 The Maneuver Automaton as a dynamical system

What we would like to remark at this point is that the Maneuver Automaton is in its own right a dynamical system, with well defined state, control, and transition function. This system evolves on what we can call the *Maneuver Space*.

*Here Pre^{-1} is a set-valued function such that $p \in \mathrm{Pre}^{-1}(q) \Leftrightarrow q = \mathrm{Pre}(p)$.

In particular, the effect of the maneuver end map, and the maneuver-trim trajectory switching is that of giving rise to a new, discrete "time" system, where "time" now is an abstraction. We will indicate the state at different "times", that is after the execution of n maneuvers-trim trajectory sequences, by square brackets, i.e.:

$$(q,h)[i+1] = \Phi\left((q,h)[i], v[i]\right).$$

Notice how one of the control inputs (namely, the coasting time τ) corresponds to a physical time duration.

The Maneuver Automaton can be seen as a consistent hierarchical abstraction of the continuous dynamics, in the sense outlined in [103]: any sequence of trajectory primitives generated by the Maneuver Automaton results by construction in a trajectory which is executable by the full continuous system. We will give a deeper meaning to hierarchical consistency in the next chapter, taking into account external disturbances.

However, we have given up all of the "nice" smoothness properties of the continuous mechanical system to develop a formulation that hinges on discrete evolution and jump relations. While this forces us to abandon tools based on smoothness and continuity of the dynamical system (such as the variational approach for optimal control), nonetheless the computational advantages of such a formulation for solving the motion planning problem can be substantial, as will be shown in Section 3.3.

Before we proceed with motion planning algorithms, we must make sure that the new dynamical system arising from the Maneuver Automaton is indeed well behaved, and maintains fundamental, desirable characteristics of the continuous model, such as well-posedness and controllability.

3.2.1 Well-posedness

One of the first questions that arises for systems governed by differential equations with discontinuities and jumps in the right-hand side (i.e. the derivatives of the state variables) is the existence and uniqueness of trajectories. If trajectories do indeed exist and are unique for any given choice of initial conditions, the system is said to be well-posed.

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 is a finite number of switches, or discontinuities in the feedback map.

3.2.2 Controllability

In this section we examine the controllability properties for the Maneuver Automaton, and give necessary and sufficient conditions under which controllability is ensured. Let us define first what we mean by controllability in the Maneuver Automaton case.

Definition 3.5 (Controllability) *We say that a maneuver automaton is controllable if, given any initial condition (q_i, h_i), and any compact set $Y \subset H$, there exists a time T such that it is possible to find an admissible sequence of primitives steering the system to any desired goal condition (q_f, h_f), with $h_f \in Y$, in time $t_f \leq T$.*

In the rest of this section we will operate on a local chart on the symmetry group H. To study controllability, we have to analyze how sequences of maneuvers, and coasting times

along trim trajectories, affect the evolution of the system. Consider a sequence of maneuvers and trim trajectories described by $v[i] = (p, \tau)[i]$, for $i = 1, \ldots, l$:

$$(q, h)[l+1] = \Phi_{v[l]} \circ \Phi_{v[l-1]} \circ \ldots \Phi_{v[1]}(q, h)[1].$$

We want to analyze what happens if we perturb the coasting times $\tau[i]$. Since we are not changing the sequence of discrete transitions, the final value of the discrete state $q[l+1]$ will not be changed. On the other hand, the final position on the symmetry group will change. We can focus on the relevant state and control variables by writing the following transition function:

$$h[l+1] = M_{\bar{v}}(h[1], \delta\tau) \tag{3.2}$$

where $\bar{v}$, indicates the sequence of nominal hybrid controls, and $\delta\tau$ indicates the sequence of perturbations on the coasting times. Keep in mind that the perturbations need not be small. The map $M_{\bar{v}}$ is continuous in its arguments: this is a consequence that the evolution of the system along trajectory primitives, and trim trajectories in particular, is continuous with respect to time. In addition, the restriction $M_{\bar{v}}(\cdot, 0)$ is invertible with a continuous inverse, and maps open sets into open sets, since it is simply a translation along the symmetry group H. For the same reason, all controllability results we will obtain are uniform over H, because of the symmetry of the system.

At this point, we have a new nonlinear, discrete-time system, in which the control inputs are the *perturbations in coasting times* along trim trajectories. Notice that if $M_{\bar{v}}(\cdot, 0)$ is a fixed-point map, that is if $M_{\bar{v}}(h, 0) = h$, then the system described by Eq.(3.2) is *driftless.* Given a maneuver sequence $\bar{v}$, we can define a reachable set under perturbations in coasting times, in the following way:

$$\mathcal{R}_{\bar{v}}(h_0, \leq T) := \left\{ h \in H \mid h = M_{\bar{v}}(h_0, \delta\tau),\ \sum_{i=1}^{l} \left(T_{p[i]} + \tau[i] + \delta\tau[i] \right) < T \right\}.$$

Now we are ready to state the following theorem:

Theorem 3.1 (Maneuver Automaton Controllability) *Consider a Maneuver Automaton as given in Definition 3.4. The Maneuver Automaton is controllable if and only if the following conditions are satisfied:*

- *The directed graph describing the discrete transitions on the automaton states is connected.*
- *There exists a (fixed point) maneuver sequence $\bar{v} = (\bar{p}, \bar{\tau})$ such that:*

$$h \in \mathrm{int}\mathcal{R}_{\bar{v}}(h, \leq T),\ \ \forall T > \sum_{i=1}^{l} T_{p[i]} + \tau[i]. \tag{3.3}$$

Proof: **Necessity** — The necessity of the first condition is easily seen. Assume that the directed graph associated with the automaton discrete state is not connected, that is, the set Q_T can be partitioned into two sets Q_{T1}, Q_{T2} such that $Q_{T1} \cap Q_{T2} = \emptyset$ and there is no valid maneuver sequence $(p_1, p_2, \ldots, p_l) \in Q_M^l$ such that $\mathrm{Pre}(p_1) \in Q_{T1}, \mathrm{Next}(p_l) \in Q_{T2}$. (Notice that there could be a maneuver sequence that takes the system from a trim in Q_{T2} to a trim in Q_{T1}.) Choose $q_i \in Q_{T1}$ and $q_f \in Q_{T2}$. Then it is impossible to find any sequence of maneuvers that satisfies the required boundary conditions.

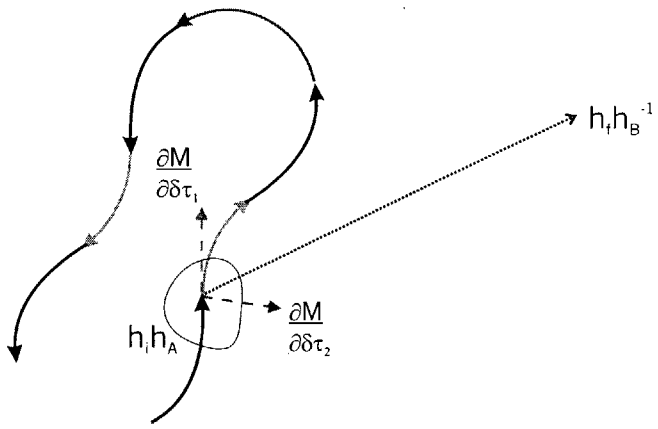

Figure 3-4: Automaton Controllability.

The necessity of the existence of a fixed point maneuver sequence is also easily proven by contradiction, by breaking the problem into two steps. Assume that the system is controllable, but there is no fixed point maneuver. Since the system is controllable, it is possible to find a sequence of maneuvers $\bar{v}_1$ to steer it from h_i to h_f in finite time, for any $h_i, h_f \in H$. After the first step is done, it must be possible to find another sequence of maneuvers $\bar{v}_2$ to steer it from h_f to h_i, again in finite time. But the composition of the maneuver sequences $\bar{v}_1$ and $\bar{v}_2$ is indeed a fixed-point maneuver.

The last step is proving that there must exist a single fixed point maneuver $\bar{v}$ such that the corresponding reachable set under coasting time perturbations satisfies (3.3). Again, we will proceed by contradiction. Assume that the system is controllable, but that there is no fixed point sequence with a reachable set with a non-empty interior. Consider a compact set $Y \subset H$, with a non-empty interior. Since the system is controllable, it will be possible to find a finite time T_1 such that any point in Y will be reachable through at least one maneuver sequence $\bar{v}_1$, of duration $T_v \leq T_1$. Because of controllability (see the previous point) , for any maneuver sequence $\bar{v}_1$, it will be possible to find a second maneuver sequence $\bar{v}_2$ such that the composition of the two sequences results into a fixed-point sequence. Since we assumed that there is no fixed-point sequence with a reachable set with a non-empty interior, and any maneuver sequence maps open sets into open sets, we must conclude that any maneuver sequence $\bar{v}_1$ has a reachable set with an empty interior (or, in other words, the reachable set has dimension less than the symmetry group H).

The number of all the possible sequences of maneuver indices $\bar{p}$ which result in a total sequence duration smaller that T_1 is finite, since each maneuver has a finite time duration, and the number of maneuvers is finite. This, however, results in a contradiction, since it is not possible to cover Y (a set with a non-empty interior) with a finite number of sets with empty interiors.

Sufficiency — Assume that there is a valid, fixed-point maneuver sequence $\bar{v}_0$, such that $\text{Pre}(\pi_0[1]) = \text{Next}(\pi_0[l]) = q_0$, and with a reachable set with a non-empty interior (i.e. satisfying Eq. (3.3)). Since the discrete evolution is described by a connected graph, it is possible to find a maneuver sequence v_A that takes the system from the initial trim q_i to the trim q_0; the maneuver sequence will result in a displacement h_A (i.e. after its execution the system is transferred from the initial conditions (q_i, h_i) to the conditions $(q_0, h_i h_A)$). Correspondingly, it is possible to find a maneuver sequence $\bar{v}_B$ that takes the system from

the trim q_0 to the final trim q_f; the maneuver sequence results in a displacement h_B. The problem of finding a finite sequence of maneuvers to steer the system from (q_i, h_i) to (q_f, h_f) is now reduced to the problem of finding a finite sequence of maneuvers steering the system from $(q_0, h_i h_A)$ to $(q_0, h_f h_B^{-1})$. Since we are operating on a local chart, it is possible to draw a line from $h_i h_A$ to $h_f h_B^{-1}$ on the chart. If the reachable set $\mathcal{R}_{\bar{v}_0}$ has a non-empty interior (which will be true for all initial conditions in H, because of the symmetry property), then it will be possible to move along the line by selecting sufficiently small steps (see Fig. 3-4). ■

Given a fixed point maneuver sequence $\bar{v}$, we are interested in computational tests to ensure that its reachable set has a non-empty interior. This can be checked simply by checking for the rank of the Jacobian matrix $\partial M(h, \delta\tau)/\partial\delta\tau$, computed at $\delta\tau = 0$, which must be the same as the dimension of H. However, we are also interested in studying what happens in the case in which a new maneuver sequence, which we will indicate by $\bar{v}^N$, is built by reiterating $\bar{v}$ a "sufficiently large", but finite, number of times N. In this case, we get the following version of the Lie algebra rank condition [131] to check that the reachable set $\mathcal{R}_{\bar{v}^N}$ has a non-empty interior:

Theorem 3.2 *The reachable set $\mathcal{R}_{\bar{v}^N}(h, \leq T)$ has a non-empty interior for all $T > \sum_{i=1}^{l} T_{p[i]} + \tau[i]$ and some $N > 0$, if $\tau_i > 0$ for all $i = 1, \ldots, l$, and the involutive closure $\bar{\Delta}$, under the Lie bracket operation, of the distribution:*

$$\Delta := \text{span}\left\{\frac{\partial M(h, \delta\tau)}{\partial\delta\tau_1}, \frac{\partial M(h, \delta\tau)}{\partial\delta\tau_2}, \ldots, \frac{\partial M(h, \delta\tau)}{\partial\delta\tau_l}\right\}$$

is such that $\dim\bar{\Delta} = \dim H$.

Proof: The system $h[l+1] = M_{\bar{v}}(h[1], \delta\tau)$ is an invertible, locally (in the "controls" $\delta\tau$) analytic nonlinear discrete-time system. The result follows from the application of Theorems 3 and 9 in [50]. ■

In the particular case of vehicles for which the symmetry group is composed by translations and rotations about a vertical axis, we can state the following:

Corollary 3.1 (Automaton controllability for $H = SE(2) \times \mathbb{R}$) *Assume that the system dynamics are invariant to translation and rotations about a vertical axis, i.e. that $H = SE(2) \times \mathbb{R}$ 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 $V_1\dot{\psi}_2\cos\gamma_1 \neq V_2\dot{\psi}_1\cos\gamma_2$, $V_1\dot{\psi}_2\sin\gamma_1 \neq V_2\dot{\psi}_1\sin\gamma_2$, and $V_1\sin\gamma_1 < 0 < V_2\sin\gamma_2$. Moreover, this is the minimum set of trajectory primitives which makes the automaton controllable.*

Proof: To prove that two trim trajectories, along with two maneuvers connecting them, are sufficient for controllability, we will show how to construct a fixed-point maneuver sequence with a non-empty interior. Without loss of generality, assume that the initial conditions are at the trim trajectory $q = 1$, at the identity on the symmetry group.

To construct a fixed-point maneuver, we proceed in the following way. Start the maneuver sequence by coasting for τ_1 seconds along the trim trajectory $q = 1$, then switching to trim trajectory $q = 2$, and coasting τ_2 seconds before transitioning back to the first trim

trajectory. We want to choose τ_1 and τ_2 in such a way that at the end of this sequence the heading is reversed, and the altitude is equal to the starting altitude. If we accomplish this, reiteration of the above sequence will take the system back to the initial condition.

This is possible if the following linear system has a meaningful solution (that is, if the solution is such that both τ_1 and τ_2 are non-negative):

$$\begin{aligned} \dot{\psi}_1 t_1 + \dot{\psi}_2 t_2 &= (2k+1)\pi - \Delta\psi_{12} - \Delta\psi_{21} \\ V_1 t_1 \sin\gamma_1 + V_2 t_2 \sin\gamma_2 &= -\Delta z_{12} - \Delta z_{21} \end{aligned} \tag{3.4}$$

where we indicate with $\Delta\psi_{12}$, $\Delta\psi_{21}$ and Δz_{12}, Δz_{21} respectively the heading and altitude changes due to maneuvers (in a body frame perspective, i.e. starting from the identity on the symmetry group).

The system (3.4) admits a solution for arbitrary maneuvers provided that its determinant is non-zero, that is

$$V_1 \dot{\psi}_2 \sin\gamma_1 \neq V_2 \dot{\psi}_1 \sin\gamma_2.$$

Moreover, this solution will be meaningful (assuming arbitrary maneuvers), in the sense that both coasting times will be non-negative, if

$$V_1 \sin\gamma_1 < 0 < V_2 \sin\gamma_2.$$

Without loss of generality we can assume that $\dot{\psi}_1 \neq 0$, and that the sideslip angle β is zero along both trim trajectories. Recalling Eqs. (2.12) and (2.11), the displacement on the symmetry group due to the first trim trajectories and the first maneuver can be expressed respectively as:

$$h_1(\tau) = \exp(\eta_1 \tau) = \begin{bmatrix} \cos(\dot{\psi}\tau) & -\sin(\dot{\psi}_1\tau) & 0 & \rho_1 \sin(\dot{\psi}_1\tau) \\ \sin(\dot{\psi}_1\tau) & \cos(\dot{\psi}_1\tau) & 0 & \rho_1 - \rho_1\cos(\dot{\psi}_1\tau) \\ 0 & 0 & 1 & V_1\tau\sin\gamma_1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(where $\rho_1 = v_1 \cos\gamma_1/\dot{\psi}_1$) and

$$h_{12} = \begin{bmatrix} \cos\Delta\psi_{12} & -\sin\Delta\psi_{12} & 0 & \Delta x_{12} \\ \sin\Delta\psi_{12} & \cos\Delta\psi_{12} & 0 & \Delta y_{12} \\ 0 & 0 & 1 & \Delta z_{12} \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Similar expressions hold for the second trim trajectory and maneuver. When the system gets back to the first trim trajectory, its position on the symmetry group will be

$$h_s(\tau_1, \tau_2) = h_1(\tau_1) \cdot h_{12} \cdot h_2(\tau_2) \cdot h_{21}. \tag{3.5}$$

If we denote by $\bar{\tau}_1$ and $\bar{\tau}_2$ the solution to the system (3.4), we have that:

$$h_s(\bar{\tau}_1, \bar{\tau}_2) = \begin{bmatrix} -1 & 0 & 0 & \Delta x_s \\ 0 & -1 & 0 & \Delta y_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

where:

$$\begin{aligned}\Delta x_s &= \rho_1 \sin(\dot{\psi}_1\tau_1) + \rho_2\left[\sin\Delta\psi_{21} - \sin(\Delta\psi_{12}+\dot{\psi}_1\tau_1)\right] \\ &\quad -\Delta x_{21}\cos\Delta\psi_{21} - \Delta y_{21}\sin\Delta\psi_{21} + \Delta x_{12}\cos(\dot{\psi}_1\tau_1) - \Delta y_{12}\sin(\dot{\psi}_1\tau_1)\end{aligned}$$

and

$$\begin{aligned}\Delta y_s &= \rho_1\left[1-\cos(\dot{\psi}_1\tau_1)\right] + \rho_2\left[\cos(\Delta\psi_{21}) + \cos(\Delta\psi_{12}+\dot{\psi}_1\tau_1)\right] \\ &\quad +\Delta x_{21}\sin\Delta\psi_{21} - \Delta y_{21}\cos\Delta\psi_{21} + \Delta x_{12}\sin(\dot{\psi}_1\tau_1) + \Delta y_{12}\cos(\dot{\psi}_1\tau_1).\end{aligned}$$

Simple algebra, or simple geometric considerations, show that reiteration of such a sequence yields a fixed-point maneuver sequence, i.e. $h_s(\bar{\tau}_1,\bar{\tau}_2)^2 = e_H$.

At this point, we need to analyze the effect of small perturbation on the (four) coasting times, that is we want to analyze the Jacobian of the map $M(h,\delta\tau) = h\cdot h_s(\bar{\tau}_1+\delta\tau_1,\bar{\tau}_2+\delta\tau_2)\cdot h_s(\bar{\tau}_1+\delta\tau_3,\bar{\tau}_2+\delta\tau_4)$ with respect to the coasting time perturbations $\delta\tau$. Notice that each of the partial derivatives $\partial M(e_H,\delta\tau)/\partial\delta\tau_i$ is a vector field at the identity on the symmetry group H, and hence is an element of the Lie algebra $\mathfrak{h}$.

In order to simplify the calculations, we can use the properties of the matrix exponential, i.e.

$$\begin{aligned}\left.\frac{\partial M(e_H,\delta\tau)}{\partial\tau_1}\right|_{\delta\tau=0} &= \left.\frac{\partial h_1(\bar{\tau}_1+\delta\tau_1)}{\partial\delta\tau_1}\right|_{\delta\tau_1=0}\cdot h_{12}\cdot h_2(\bar{\tau}_2)\cdot h_s(\bar{\tau}_1,\bar{\tau}_2) \\ &= \left.\frac{\partial\exp(\eta_1\tau)}{\partial\tau}\right|_{\tau=\bar{\tau}_1}\cdot h_1(\bar{\tau}_1)^{-1}\cdot M(e_H,0) = \eta_1 = \eta_1\cdot h_1(\bar{\tau}_1)\cdot h_1(\bar{\tau}_1)^{-1} = \eta_1.\end{aligned}$$

Similar considerations can be made for the other partial derivatives; In the case in which h_{21} is equal to the identity (e.g. if the vehicle can transition instantaneously from one trim trajectory to the other), we also have

$$\frac{\partial M(e_H,\delta\tau)}{\partial\delta\tau_4} = \eta_2.$$

Making use of Theorem 3.2, we can check controllability of the Maneuver Automaton by ensuring that the involutive closure of the distribution $\mathrm{span}\{\eta_1,\eta_2\}$ under the Lie bracket operation has dimension four. This is the case if

$$\dot{\psi}_1(\dot{\psi}_1 v_2\cos\gamma_2 - \dot{\psi}_2 v_1\cos\gamma_1)^2(\dot{\psi}_1 v_2\sin\gamma_2 - \dot{\psi}_2 v_1\sin\gamma_1) \neq 0, s \tag{3.6}$$

which is true under the first two conditions stated in the theorem.

In the general case, tedious but straightforward calculations yield that the vector fields $\partial M(e_H,\delta\tau)/\partial\delta\tau_i$ spans the Lie algebra $\mathfrak{h}$ under almost all arbitrary maneuvers, if condition (3.7) is satisfied. In the case in which the maneuvers are such that the Jacobian matrix loses rank, it is sufficient to repeat the same construction in this proof, but replacing π by π/m in the first equation of system (3.4) for some integer m, and considering the m-th power of h_s when constructing the fixed-point maneuver.

To prove the last claim of the corollary, we can show that any automaton with only one trim trajectory, and a finite number of maneuvers (to and from the same trim trajectory),

is not controllable. This is readily seen by examining the evolution of the altitude and of the heading. We have that for any finite maneuver sequence

$$\Delta z = \sum_{i}^{n} \left(v_1 \sin \gamma_1 \tau_i + \Delta z_i \right),$$

and

$$\Delta \psi = \sum_{i}^{n} \left(\dot{\psi}_1 \tau_i + \Delta \psi_i \right).$$

All the partial derivatives with respect to coasting times, for any maneuver sequence will have the form $[v_1 \sin \gamma_1, \dot{\psi}_1]^T$, and give rise to a one-dimensional distribution (clearly closed under the Lie bracket operation). This means that it is not possible to control independently attitude and heading with only one trim trajectory in the maneuver automaton. ∎

Corollary 3.2 (Automaton Controllability for $H = SE(2)$) *If the motion of the vehicle is restricted to a horizontal plane ($\gamma_1 = \gamma_2 = 0$), then the vehicle is controllable if $V_1 \dot{\psi}_2 \neq V_2 \dot{\psi}_1$.*

Proof: The proof is the same as the proof of Corollary 3.1, with the only difference that there are no translations about the vertical axis. As a consequence, the system (3.4) reduces to its first equation only, and the only condition needed to build a maneuver sequence h_s is that at least one of the turning rates is non-zero. The condition 3.2 translates to

$$\dot{\psi}_1 (\dot{\psi}_1 v_2 - \dot{\psi}_2 v_1)^2 \neq 0, \tag{3.7}$$

which requires that the two trim trajectories have different signed turning radii. The same requirement ensures controllability in the case of arbitrary maneuvers. ∎

Notice that if the maneuvers are instantaneous (i.e. the vehicle can transition instantaneously between two different trim trajectories), Corollary 3.2 reduces to a version of stronger results obtained by Dubins [31] and Reeds and Shepp [108] concerning optimal paths for kinematic cars on the plane (see also [127]). However, the author is not aware of results similar to Corollary 3.1.

It is clear that the sets of trim trajectories in the two corollaries above would be just a minimum set to ensure controllability: For practical applications, the set of trim trajectories will be much richer. Notice that a ground vehicle which can only turn left with different turning radii (as in [71]), is controllable according to our definition, even though it is not small-time controllable [130].

3.3 Motion planning in the Maneuver Space

The hybrid control architecture lends itself to computationally efficient solutions to many problems of interest for practical applications. The price that we must pay in using the maneuver 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.

By the structure of the Maneuver Automaton, it is also evident that its applicability is limited to problems in which the target state (and, to a more limited extent, the initial conditions) correspond to trim trajectories. This is not a very limiting requirement, since in fact most meaningful motion planning problems are stated exactly in these terms. For example, a typical requirement for a helicopter could be to stop or hover at a certain location, altitude and heading. The target roll angle and pitch heading, and other state variables, are not directly specified, and will assume the values required to achieve the desired steady hovering conditions.

3.3.1 Optimal control in a free environment

In Section 3.2.2 we have examined the conditions under which, given a Maneuver Automaton, it is possible to find a sequence of maneuvers and coasting times along trim trajectories such that the system is steered from any initial condition to any desired final condition. Even though the proof of the sufficiency of Theorem 3.1 provides a method of constructing a trajectory satisfying the boundary conditions of Problem 1.2, the resulting trajectories are likely to be far from optimal. Also, in practical implementations the number of maneuvers and trim trajectories available in the Automaton will be much larger than the number of primitives strictly needed for controllability. As a consequence, it is desirable to characterize the quality of the generated trajectory according to some defined cost, that is, we want to formulate an efficient computation-based method for the solution of Problem 1.3.

Consider again the cost functional (1.2):

$$J(x(\cdot), u(\cdot)) := \int_{t_0}^{t_f} \gamma(x, u)\, dt,$$

and make the following assumption:

Assumption 3.3 (Invariance of the cost function) *The incremental cost $\gamma(x, u)$ is invariant with respect to group actions on the symmetry group: $\gamma(x, u) = \gamma(\Psi_h(x), u) \;\; \forall h \in H$.*

Examples of problems with invariant cost functions include minimum time, minimum length, minimum control effort. A notable exception consists of quadratic cost functions in the states: Such cost functions are invariant with respect to H only if the states which "belong" to H are not weighted.

Based on Assumption 3.3, we can derive the following expression for the cost increment along trajectories generated by the Maneuver Automaton. Along a trim trajectory q, we have (with a slight abuse of notation):

$$\frac{dJ}{dt} = \gamma(g, \xi, u) = \gamma(\bar{g}_q, \bar{\xi}_q, \bar{u}_q) =: \gamma_q.$$

In the above we used the fact that $g(t) = h_0 \exp(\bar{\eta}_q t)\bar{g}_q$. The cost increment along a trim trajectory is hence a constant, depending only on the trim trajectory being executed.

Similarly, after the execution of a maneuver p we have:

$$\Delta J = \int_0^{T_p} \gamma(\Psi_{h_0}\phi_p(t), u_p(t))\, dt = \int_0^{T_p} \gamma(\phi_p(t), u_p(t))\, dt =: \Gamma_p.$$

Again, we have that the (finite) cost increment after the execution of a maneuver is a constant, depending only on the maneuver index.

We can now formulate an expression for the cost functional (1.2) that is compatible with the maneuver automaton structure:

$$\tilde{J}(q[\cdot], h[\cdot], p[\cdot], \tau[\cdot]) = \sum_{i=1}^{n_f - 1} \gamma_{q[i]} \tau[i] + \Gamma_{p[i]}, \tag{3.8}$$

with the constraint that $(q, h)[n_f] = (q_f, h_f)$ for some finite n_f. Note that at this point Eqs. (1.2) and (3.8) are equivalent, the only difference being the different parameterization of trajectories.

Since now we have a discrete "time" system, the application of the optimality principle leads us to a version of the Bellman's equation (as opposed to the Hamilton-Jacobi-Belmann partial differential equation). Define the optimal cost function $\tilde{J}^*$, obtained by optimizing on the automaton structure (i.e. on the Maneuver Space). The optimality principle implies that the optimal cost function satisfies

$$\tilde{J}^*(q, h) = \min_{(p,\tau)} \left\{ \gamma_q \tau + \Gamma_p + \tilde{J}^* \left[\Phi_{(p,\tau)}(q, h) \right] \right\}. \tag{3.9}$$

Note that, since the trajectories generated by the Automaton are a subset of all trajectories executable by the full continuous system, the optimal costs computed on the Maneuver Automaton will be in general greater that the true optimal cost:

$$\tilde{J}^*(q, h) \geq J^*(h\bar{g}_q, \bar{\xi}_q).$$

On the other hand, the solution to the Bellman's equation (3.9) is, from the computational point of view, much simpler than the solution of the Hamilton-Jacoby-Bellman partial differential equation (1.7). The hybrid state space $Q_T \times H$ (the Maneuver Space) is much smaller that the continuous state space $G \times \mathfrak{g}$. For the general vehicle case (as seen in Chapter 2), the hybrid state has dimension four — consisting of three position and one heading coordinates — while the continuous state has dimension at least twelve. Moreover, the decision variable set in the Automaton case is $Q_M \times \mathbb{R}_+$, which is much smaller than the continuous control set $\mathcal{U}$ (in the case of aircraft, usually at least four).

If the dimension of the symmetry group is still too large for computing the optimal cost function, it can be possible to further reduce the computational complexity of the problem by decomposing the group H in smaller symmetry groups (possibly of dimension one), and solve the optimal control problem in each subgroup in sequence. Naturally, such an approach would reduce the optimality of the resulting trajectory (essentially, this amounts to working on one coordinate at a time).

If the optimal cost function $\tilde{J}^*(q, h)$ is known, the optimal control can be computed by solving

$$(p, \tau)^* = \arg\min_{(p,\tau)} \left\{ \gamma_q \tau + \Gamma_p + \tilde{J}^* \left[\Phi_{(p,\tau)}(q, h) \right] \right\}, \tag{3.10}$$

which is an optimization problem on a discrete variable $p \in Q_M$, and a continuous, scalar variable $\tau \in \mathbb{R}_+$ (and as a consequence can be solved as a sequence of scalar optimization problems).

Thus the formulation of the optimal control problem on the "Maneuver Space" can be seen as a trade-off between optimality and computational complexity. In the following (see Section 3.4.1) we will further characterize this trade-off, for the time being we will proceed with the discussion of the computation of the (sub)-optimal solution.

3.3.2 Computation of the optimal cost

Since the dimension of the hybrid state is relatively small (in the cases we are considering, it is never more than four, see Section 2.3.4), it is possible to apply efficient computational methods developed for approximate dynamic programming, also known as neuro-dynamic programming [10], and reinforcement learning in the artificial intelligence community. The main approaches to the solution of dynamic programming problems are based on the computation of the optimal cost J^* (Value Iteration), of the optimal control policy $\mu^* : (q,h) \mapsto (p,\tau)^*$ (Policy Iteration), or of the so-called *Q-factors*, i.e. the functions to be minimized in Eqs. (3.10) and (3.9) (Q-learning).

We will now briefly present the basic algorithm of Value Iteration. We refer the reader to [10], for more information and details on the Value Iteration algorithm and other computational methods for approximate dynamic programming.

Value Iteration based on a greedy policy

Assume we have an initial estimate of the cost function $J_0(q,h)$, which is an upper bound on the optimal cost $J^*(q,h)$ (we will drop the tilde from now on, as long as no confusion with the optimal cost for the continuous system can arise). Such an initial estimate can be obtained from simple heuristics, by the solution of a feasibility problem (e.g. through the construction of trajectories as in the proof of Theorem 3.1), or even be set to $J_0(q,h) = \infty$ for $(q,h) \neq (q_f,h_f)$, and $J_0(q_f,h_f) = 0$.

Starting from the initial estimate J_0 it is possible to implement an iterative algorithm, as follows. At step i a no worse approximation of the optimal cost function is given by the following update:

$$J_i(q,h) = \min_{(p,\tau)} \left\{ \gamma_q \tau + \Gamma_p + J_{i-1}\left[\Phi_{(p,\tau)}(q,h)\right] \right\}.$$

Clearly, $J^*(q,h) \leq J_i(q,h) \leq J_{i-1}(q,h)$. Under the assumption that we visit *each state* infinitely often, and other technical conditions [11], this iteration would converge to the optimal cost function $J^*(q,h)$, in the sense that, as $i \to \infty$, $\sup_{(q,h)} [J_i(q,h) - J^*(q,h)] \to 0$.

However, since h is a continuous variable, it is not possible to visit each state infinitely often; moreover, storage of the function $J_i(q,h)$, requires in general some form of discretization.

Value Iteration with Representative States

In order to deal with the infinite cardinality of the set H, we consider a grid on H, as a finite set of points, or representative states, h_i, $i = 1 \ldots K$, and introduce the following approximation architecture:

$$\hat{J}(q,h) = \sum_{i=1}^{K} \lambda_i(h) \hat{J}^k(q) \tag{3.11}$$

where the coefficient functions are such that $\sum_{i=1}^{K} \lambda_i(h) = 1$, and $\lambda_i(h_j) = \delta_{ij}$ (where δ_{ij} is the Kroenecker delta). The coefficients $\lambda_i(h)$ can be seen as a measure of the closeness of the state h to the representative state h_i.

In particular, we will consider a piecewise-linear approximation, built in the following way. Choose a grid in which the representative states partition the set H into a collection of simplexes, with disjoint interiors. Then, any state h will lie inside one simplex, with

vertices $h_{i_1}, h_{i_2}, \ldots, h_{i_{(n+1)}}$ (where n is the dimension of H) and it will be possible to find nonnegative coefficients λ_{i_j} such that (in local coordinates):

$$h = \sum_{j=1}^{n+1} \lambda_{i_j} h_{i_j}.$$

Correspondingly, the approximation of the cost function at h will be given by:

$$\hat{J}(q,h) = \sum_{j=1}^{n+1} \lambda_{i_j} \hat{J}^{i_j}(q).$$

Under technical conditions elicited in [10, pages 341-364], satisfied in our case, an incremental value iteration algorithm based on Eq.(3.11) will converge to an estimate $\hat{J}_\infty$ of the cost function J^*. A measure of the power of the approximation architecture is given by the error ϵ introduced by the approximation, where $\epsilon = \max_{q,h} \|\hat{J}(q,h) - J^*(q,h)\|$. It has been proven that there exists some $\epsilon_0 > 0$ such that, if $\epsilon < \epsilon_0$, a greedy policy based on $\hat{J}$ will be optimal [11].

The objective of the off-line computation phase will then be the construction of an approximation to the optimal cost function, such that the on-line implementation of a greedy policy will result in an optimal control strategy.

3.4 Examples

In this section we examine a few examples of the construction of Maneuver Automata and their use. The first example deals with a system which is simple enough to compute trim trajectories, maneuvers, and optimal cost from the model. The second example shows how to construct a maneuver automaton from experimental data. The third example involves model-based construction of trajectory primitives and computation of the approximate optimal cost function.

3.4.1 Double integrator

As a first example, we will look at the simplest mechanical system, namely the double integrator. The dynamics are given by $\ddot{y} = u$; furthermore, we will impose the "flight envelope" constraint $\|\dot{y}\| \leq v_{max}$ and the control saturation constraint $|u| \leq u_{max}$.

Trim trajectories are given by all trajectories along which $u = \bar{u}_i = 0$, and $\dot{y} = \bar{\eta}_i$, $i \in Q_T$. Maneuvers are designed as minimum-time transitions between two trim trajectories, defined by a constant control input $u_{ij} : [0, T_{ij}) \to \{\text{sign}(\bar{\eta}_j - \bar{\eta}_i)\}$, and $T_{ij} = |\bar{\eta}_i - \bar{\eta}_i|$, $i, j \in Q_T$ (see Section 3.1.1).

Necessary and sufficient condition for controllability of a Maneuver Automaton composed of a finite number of the above primitives is that there exist $i, j \in Q_T$ such that $\bar{\eta}_i \bar{\eta}_j < 0$.

Consider now the problem of steering the double integrator to the origin, in minimum time. The incremental cost function is thus $\gamma(y, \dot{y}, u) = 1$; the optimal control solution can be derived by the analysis of the behavior of the system on the phase plane $(y, \dot{y})$ (Fig. 3-5).

Introducing the switching function:

$$\Delta_0(y, \dot{y}) = \begin{cases} \dot{y}^2 + 2u_{max}y & \text{for } \dot{y} \geq 0 \\ -\dot{y}^2 + 2u_{max}y & \text{for } \dot{y} < 0 \end{cases}$$

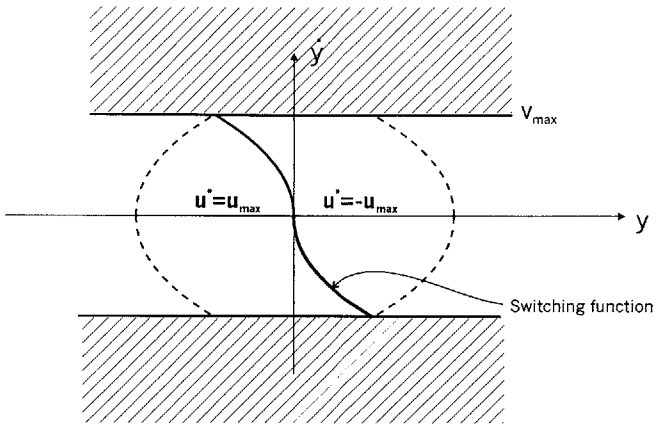

Figure 3-5: Phase plane regions for the double integrator.

we can distinguish two main distinct regions in the phase space. In the interior of the leftmost region in Fig. 3-5, the optimal control is equal to u_{max}, while in the interior of the rightmost region, the optimal control is equal to $-u_{max}$. On the boundary corresponding to the switching function, the optimal control is equal to $-u_{max}\,\mathrm{sign}\,\dot{y}$. Finally, on the boundary corresponding to the maximum velocity constraint, the optimal control is zero.

The optimal cost (minimum time to the origin) in the case in which the velocity does not saturate can be computed as:

$$J^*(y,\dot{y}) = t_f^*(y,\dot{y}) = t_1(y,\dot{y}) + t_2(y,\dot{y}),$$

where t_1 is the "acceleration" time, and t_2 consists of the "deceleration" time along the switching function:

$$\begin{aligned} t_2(y,\dot{y}) &= \frac{\sqrt{1/2\dot{y}^2 - yu^*}}{u_{max}} \\ t_1(y,\dot{y}) &= t_2(y,\dot{y}) - \frac{\dot{y}}{u^*}. \end{aligned}$$

If the velocity reaches its maximum allowed value, the minimum time is given by acceleration and deceleration times, together whit a coasting time, corresponding to the time interval during which the magnitude of the velocity attains its maximum value. Velocity saturation will eventually occur if the initial conditions are "far enough" from the origin . The regions corresponding to initial conditions that will lead to velocity saturation are delimited by the curves

$$\Delta_{\pm 1}(y,\dot{y}) = (\dot{y}^2 - 2v_{max}^2) \pm 2u_{max}y = 0,$$

which are depicted as dashed lines in Fig. 3-5. In the leftmost region:

$$t_f^*(y,\dot{y}) = \frac{\Delta_{-1}(y,\dot{y})}{2u_{max}v_{max}} + \frac{v_{max} - \dot{y}}{u_{max}} + \frac{v_{max}}{u_{max}}.$$

For initial conditions at rest, the optimal cost function is proportional to the square root of the distance to the origin, until the point x_s at which the initial conditions are such that the velocity will reach its maximum magnitude. For initial condition farther away from the origin, the optimal cost will increase linearly, with a slope inversely proportional to v_{max} (see Fig. 3-6).

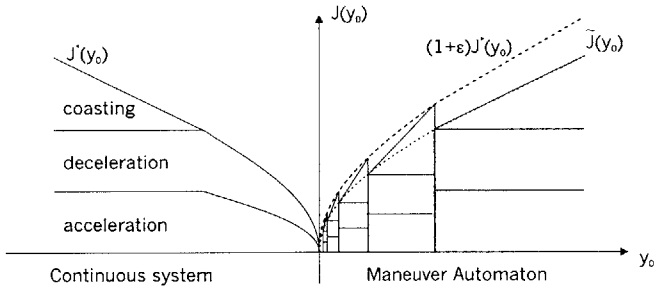

Figure 3-6: Minimum-time optimal cost for double integrator with initial conditions at rest.

The cost function we have computed so far corresponds to the *exact* solution to the minimum-time problem on the constrained double integrator. Now we want to study what happens if we constrain the system trajectories to be trajectories generated by a Maneuver Automaton. To this purpose, consider first an automaton composed by the following trim trajectories: $\bar{\eta}_0 = 0$ (rest condition), and $\bar{\eta}_{\pm 1} = \pm v_{max}$ (maximum velocity, in the positive and negative directions), and minimum-time maneuvers connecting them. For initial conditions sufficiently far from the origin, the optimal cost $\tilde{J}_0^*$ for the system generated by the Maneuver Automaton is *exactly the same* as the true optimal cost for continuous system J^*. However, as the initial condition gets closer to the origin, the optimal solution will no longer saturate the velocity: thus switching among the trim trajectories in the Automaton is no longer optimal. The effect of the quantization of maneuvers is that the system will have to overshoot, and get far enough on the opposite side of the origin so that an additional velocity saturating sequence composed of acceleration and deceleration maneuvers, leading exactly to the origin, can be accomplished. Similar effects can be noticed on the optimal cost given initial conditions at maximum velocity. The optimal cost for the Maneuver Automaton $\tilde{J}_0^*$ is in such cases much higher than the true cost J^*; in other words, the optimality penalty we must accept for the use of the Automaton framework is relatively very high.

Optimal selection of trajectory primitives

The question that we want to address at this point is the following. Given a maximum number of trajectory primitives, what is the choice of trim trajectories such that the optimal control solution on the Maneuver Automaton can approximate the "true" solution to the maximum extent? In other words, what is the choice of primitives which minimizes $\tilde{J}^* - J^*$? Conversely, we are interested in the question: Given a desired optimality gap, can we characterize the minimum number of trajectory primitives that achieves it? In the following we will present the solution to these problems in the simple case of the double integrator. Even though the extensibility of the results to more general systems is not clear at present, the results will provide some important insight in the relationship between trajectory primitive selection and optimality.

The first result addresses the problem of finding the coarsest, countable quantization of trim trajectories, which results in a given penalty ϵ on the optimal cost, for initial conditions at rest (we are considering possibly infinite, but countable, sets of trajectory primitives). By coarsest here we mean the quantization such that the number of different trim trajectories with velocity setting between v_{max} and any $v_{min} \in (0, v_{max})$ is minimized.

Theorem 3.3 (Optimal minimum-time quantization) *Given a tolerance $\epsilon > 0$ on*

the cost penalty, the coarsest trajectory primitive quantization, such that $\tilde{J}^*(0,y) \leq (1+\epsilon)J^*(0,y)$*, is given by the countable set of trim trajectories:*

$$\bar{\eta}_{\pm i} = \pm\rho^{i-1}v_{max} \tag{3.12}$$

where:

$$\rho := 1 + 2\epsilon - 2\sqrt{\epsilon^2+\epsilon},$$

together with the countable set of minimum-time maneuvers between all pairs of trim trajectories.

Proof: We are considering the baseline case in which the initial conditions are at rest, i.e. $q = 0$, $\bar{\eta}_0 = 0$. Defining $Y := \frac{v_{max}^2}{u_{max}}$, the optimal cost function in this case is given by:

$$J^*(0,y) = \begin{cases} \sqrt{\frac{|y|}{u_{max}}} & \text{for } |y| < Y \\ \frac{|y|}{v_{max}} - \frac{v_{max}}{u_{max}} & \text{for } |y| \geq Y \end{cases}.$$

As we already know, the cost function $\tilde{J}_0^*(0,y) = J^*(0,y)$ for $|y|$ sufficiently large, that is for $|y| \geq Y$.

Now consider the same setup, but constrain the velocity to be bounded by ρv_{max}, with $\rho < 1$. The optimal cost function in this case will be given by:

$$J_\rho^*(0,y) = \begin{cases} \sqrt{\frac{|y|}{u_{max}}} & \text{for } |y| < \rho^2 Y \\ \frac{|y|}{\rho v_{max}} - \frac{\rho v_{max}}{u_{max}} & \text{for } |y| \geq \rho^2 Y \end{cases}.$$

As it can be expected, $J_\rho^*(0,y) \geq J^*(0,y)$. However, if we consider ρv_{max} a new level of quantization of the Maneuver Automaton trim trajectories, we have that the corresponding optimal cost function $\tilde{J}_1(0,y) < \tilde{J}_0(0,y)$ for $|y| < Y$. As a consequence, we can pick ρ in such a way that $\tilde{J}_1(0,\pm Y) = J_\rho^*(0,\pm Y) = (1+\epsilon)J^*(0,\pm Y) = (1+\epsilon)\tilde{J}_0(0,\pm Y)$. The *smallest* value of ρ that satisfies this condition can be computed as follows:

$$\rho = 1 + \epsilon - \sqrt{\epsilon^2 + 2\epsilon}.$$

Notice that this is the smallest ρ such that the cost $\tilde{J}_1$ is always bounded above by $(1+\epsilon)J^*$, for initial conditions at rest (see Fig. 3-6).

Now we can repeat the argument for the next level of quantization: since the above expression for ρ we derived only depends on ϵ, the next quantization level will be given by $\bar{\eta}_{\pm 2} = \pm\rho^2 v_{max}$, and each couple $\bar{\eta}_{\pm i}$ will provide an ϵ approximation of the true optimal cost over the interval $\rho^i Y \leq |y| < \rho^{i-1}Y$. We have thus established that a logarithmic scaling of the velocity quantization levels provides the coarsest Automaton capable of approximating the optimal cost function J^* within a factor of $1+\epsilon$, from initial conditions at rest. ∎

Now we would like to extend this result to initial conditions with non-zero velocity. That is, we would like to analyze the penalty induced by the velocity quantization on the

optimal cost, if the initial conditions are not at rest. This can be done by recognizing that the optimal cost can be written as:

$$J^*(\dot{y}, y) = J^*(0, y') - \frac{\dot{y}}{u^*(\dot{y}, y)}, \tag{3.13}$$

where $y' = y - 1/2\ \dot{y}^2/u^*(\dot{y}, y)$. The second term on the right hand side of Eq. (3.13) corresponds to the time needed to accelerate or decelerate between zero and the initial velocity, and its magnitude is less than or equal to the optimal cost.

Since the Automaton includes time-optimal maneuvers, we also have that:

$$\tilde{J}^*(\rho^i v_{max}, y) = \tilde{J}^*(0, y') - \frac{\rho^i v_{max}}{u^*(\rho^i v_{max}, y)}.$$

As a consequence:

$$\begin{aligned} \tilde{J}^*(\rho^i v_{max}, y) &\leq (1+\epsilon) J^*(0, y') - \frac{\rho^i v_{max}}{u^*(\rho^i v_{max}, y)} \\ &\leq (1+\epsilon) J^*(\rho^i v_{max}, y) + \epsilon J^*(\rho^i v_{max}, y) = (1+2\epsilon) J^*(\rho^i v_{max}, y). \end{aligned}$$

Thus, the quantization (3.12) results in a penalty on the optimal cost bounded by ϵJ^* for initial conditions at rest, and by $2\epsilon J^*$ for initial conditions with non-zero velocity.

These results, even though limited to a very simple case, do provide some interesting insight in the nature of the optimal choice of trim trajectories, or quantization levels. When far from the "equilibrium", and moving at high speed, relatively few quantization levels are needed for almost optimal performance. However, as the speed is reduced, and the importance of fine motion control becomes predominant, a much finer quantization is required. This is in accordance with the concept of "minimum attention control" [20]. In the following, we will always choose trim trajectory based on a logarithmic quantization of the trim parameter space, even though this is not a provably optimal choice in the general case.

One may note that the optimality of a logarithmic quantization scale for a double integrator bears a strikingly similarity to results obtained for quadratic stability of quantized linear systems [33], and nesting of switching sets in the so-called High Performance Bounded control methodology [121].

3.4.2 Three degrees of freedom helicopter

In this section we will address the minimum-time control problem for the three degrees of freedom helicopter introduced in section (2.3.2). The main motivation of this section is to show how a Maneuver Automaton can be built from the analysis of experimental data, together with insight in the fundamental properties of the system's dynamics. Even though a detailed model for the system could be derived (including all aerodynamic forces and rotor inflow modelling), the construction of the Maneuver Automaton for this system will be carried out with the assumption that the dynamics are not precisely known, and most of the information about the vehicle has to be derived from experiments.

As already mentioned, the symmetry group for the system is the group of rotations about a vertical axis, $H = S^1$. Correspondingly, relative equilibria correspond to constant heading rate rotations. For simplicity of exposition, we will only consider rotations with zero pitch angle (this measures the inclination of the arm with respect to the horizontal plane). Moreover, for safety reason, the roll angle ϕ is limited to ± 30 degrees.

Experimentally, it was found that the maximum speed of the helicopter, while maintaining constant altitude, and a roll angle of $\pm 30°$ is about $110°$ per second, which is reached asymptotically. All velocity settings in $(-110, 110)$ are achievable in steady state, and so the trim trajectory selection can be carried out at will in this set. As a practical maximum velocity setting we can choose $\pm 80°/s$, which is achieved in about $16.5s$ and a total rotation of about $800°$, and then choose other velocity setting using a logarithmic scaling with $\rho = 1/2$; this scaling would result, in the ideal case of a double integrator, in a performance loss of $\epsilon = 25\%$. Finally, only six velocity settings other than zero were considered, namely $(\pm 80, \pm 40, \pm 20)°/s$.

Since the dynamics are not well known, especially at high velocity regimes, the problem of determining the full trim conditions (i.e. $(\bar{g}, \bar{u})_q$, other than the imposed $\bar{\eta}_q$) is solved through the use of a feedback controller, designed to track steps in $\dot{\psi}$; the feedback control law used is a simplification of the control law for small helicopters detailed in Chapter 5. [†] The results are reported in Table 3.1.

Index $q \in Q_T$	Heading rate $\dot{\psi}[deg/s]$	Roll angle $\phi[deg]$
0	0	-1
1	20	-5
2	40	-7.5
3	80	-15
4	-20	2
5	-40	4.5
6	-80	12

Table 3.1: Trim settings for the three degrees of freedom helicopter.

The maneuvers are also generated through the same feedback control law. Strictly speaking, since the controller provides only asymptotic tracking (even if at an exponential rate), these are not *finite time* transitions from trim to trim. However, for all practical purposes it is possible to choose the maneuver termination time as the time at which the state of the system settles to within a given accuracy (depending for example on the noise level of the sensors).

Maneuvers were generated and recorded for each transition between trim trajectories (and averaged over several runs). Results, in the form of the aggregate maneuver data $(T, \Delta\psi)$ are given in Table 3.2. Note that we "unroll" the heading coordinate (we keep track of the number of revolutions: a rotation of $360°$ is different from a zero rotation); as a consequence our symmetry group is actually $\mathbb{R}$. An example of a maneuver (from $-20°/s$ to $40°/s$ is also reported in Fig. 3-7.

Once the maneuver library is available, it is desired to use it to solve in real-time minimum-time optimal control laws. The (sub)-optimal cost function can be built through the algorithm described in Section 3.3.2. However, in the case at hand, we can make good use of additional properties of the Maneuver Automaton to achieve a better approximation of the cost function.

A fact that is always true is following:

[†]It is important to remark that this feedback control law is only used for the off-line generation of trajectory primitives. While it can also be used for on-line local stabilization of the system, it is not appropriate for the solution of minimum-time control problems.

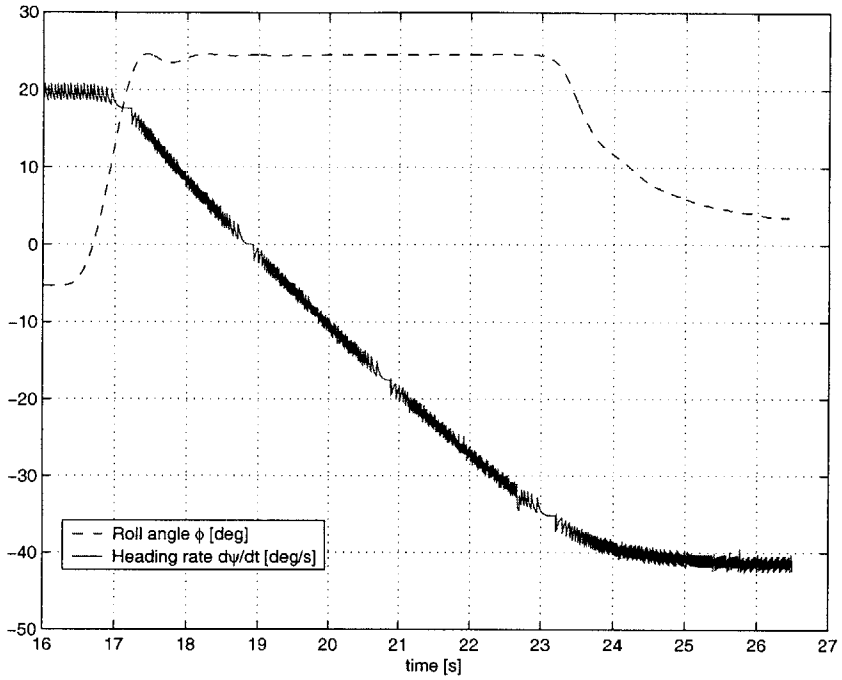

Figure 3-7: Example maneuver for the 3-DOF helicopter.

Proposition 3.1 *The gradient of the Maneuver Automaton optimal cost function, at points such that $\tau^* > 0$, satisfies:*

$$\frac{\partial \tilde{J}(q,h)}{\partial h} h\bar{\eta}_q = -\gamma_q.$$

Proof: Along an optimal trajectory the total derivative $d\tilde{J}(q(t),h(t))/dt$ of the optimal cost-to-go must be equal to the negative of the incremental cost g. Since the optimal cost function does not depend explicitly on time, along a trim trajectory (i.e. for constant q), we have that $d\tilde{J}/dt = \partial \tilde{J}(q(t),h(t))/\partial h \cdot \dot{h}$. But along a trim trajectory we also have that $\dot{h} = h\bar{\eta}_q$, and $g = -\gamma_q$, which proves the claim. ∎

This can be seen a restatement of the Hamilton-Jacobi-Bellman equation (1.7) along a trim trajectory of the Maneuver Automaton. However, the optimal cost-to-go function is not continuously differentiable in the Maneuver Automaton case — it is not even continuous — and it cannot be derived as a solution of the Hamilton-Jacobi-Bellman partial differential equation.

The result in Proposition 3.1 is especially useful when the symmetry group is one-dimensional, because in this case the gradient of $\tilde{J}$ is completely characterized. In addition, we can extend the result to non-isolated points on a switching surface (that is, points such that the optimal coasting time is equal to zero, and an immediate switch is demanded for optimality), since the gradient in such cases will be the gradient corresponding to the first trim trajectory on which the ensuing maneuver sequence will stay for a finite amount of time.

Using the above considerations on the gradient of the optimal cost function, and storing optimal controls along with the optimal cost at each "representative state", or grid point, we

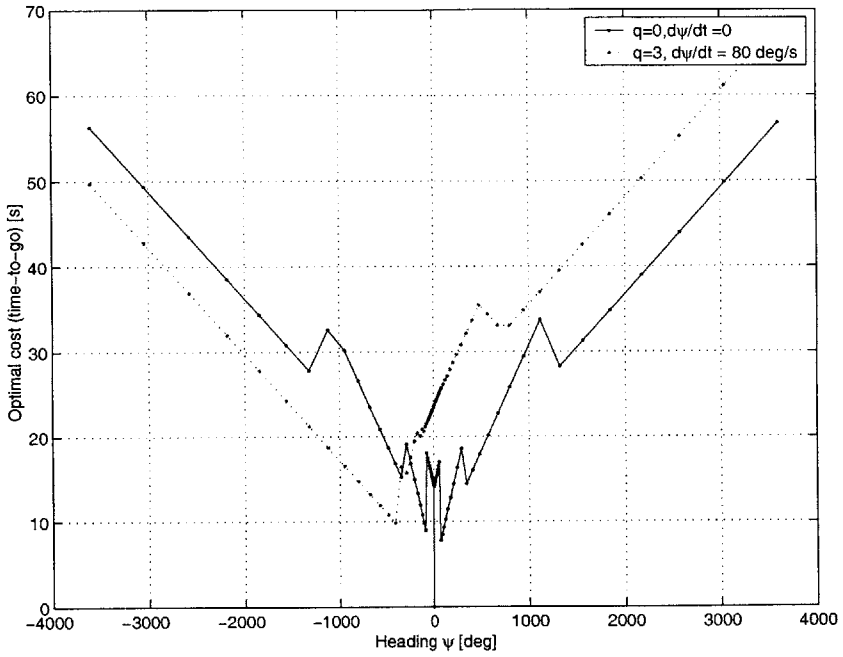

Figure 3-8: Optimal cost for the 3-DOF helicopter Maneuver Automaton.

can build an approximation architecture that performs better than the general piecewise-constant architecture in Section 3.3.2 (in the sense that the approximation error will be smaller for the same choice of representative states). The optimal cost function for initial conditions at rest and initial conditions at the maximum velocity trim setting are reported in Fig. 3-8.

Based on the Maneuver Automaton structure outlined in this section, and the corresponding optimal cost just computed, a real-time controller was implemented on a three degrees of freedom helicopter. The experiment was run as follows. First of all, a take-off phase is executed, in which the helicopter can acquire the desired state, hovering at zero pitch angle, and zero heading (that is on the vertical of the take-off position). At $t = 5s$, a command is given to hover at $-90°$. At time $t = 12s$, before the helicopter reached the previously commanded position, a new command is issued, according to which the helicopter must hover at $+360°$. The reference trajectory is generated by the maneuver automaton, and a tracking controller ensures that the the state of the helicopter stays "close" to the reference trajectory (more details on the design of feedback controllers to integrate with the Maneuver Automaton will be given in the next chapter). The recorded data from the experiment are shown in Fig. 3-9. As it can be seen from the picture, the generated sub-optimal trajectory is consistent with the system's dynamics, and can be followed very accurately by a very simple controller. Very little knowledge of the system's dynamics (at least in an analytical model) was used in generating the trajectory, and only results from experiments formed the basis of the maneuver library. In the bottom picture is also reported the evolution of the discrete state $q \in Q_T \cup Q_M$ (with the indices of the maneuvers available in Tables 3.1,3.2).

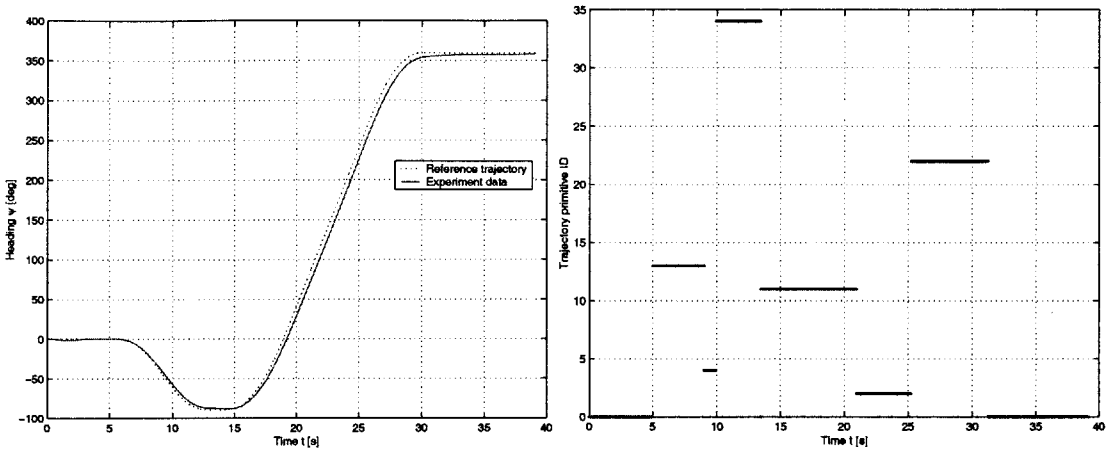

Figure 3-9: Real-time sub-optimal trajectory generation experiment for the 3-DOF helicopter.

3.4.3 Small autonomous helicopter

As a last application example, we consider the minimum time optimal control problem, for a small autonomous helicopter. We will use a mixed approach in the generation of the trajectory primitives for constructing a Maneuver Automaton.

First we will use an approximation of the helicopter dynamics to construct maneuvers. If in the model of the helicopter presented in Chapter 2 we assume that the only input force acting on the system is directed along a body-fixed direction, and that drag forces are attitude-independent, the system becomes differentially flat (see also [88, 137, 62]). A possible choice of flat outputs consists of the position and heading (that is, the position on the symmetry group H).

Differential flatness of the system means that, given any trajectory $h(t)$ and its derivatives, we can recover completely the state and control trajectories that generate it. This can be readily verified in the following way. Given a desired acceleration $\ddot{x}$, the corresponding force f might easily be derived by solving:

$$m\ddot{x} = f + m\tilde{g} + b(\dot{x})$$

where $\tilde{g}$ is the gravity acceleration, and b represents the drag forces, assumed to be attitude-independent. The magnitude of f gives us immediately the amount of thrust ($u_4 = T$) required. Since in this simplified model of the helicopter the thrust is generated along a body-fixed direction, the attitude of the helicopter is determined, up to a rotation about the direction of f. However, since ψ is given (it is one of the flat outputs), the attitude can be uniquely determined using this additional piece of information[‡]. By looking at up to the fourth-order derivative of the position, we can eventually compute the required angular acceleration, and compute the required control torques u_1, u_2, u_3.

It is then immediate to compute the trim conditions, assigning $h(t) = \exp(\bar{\eta}_q t)$. In order to compute maneuvers, a convenient approach is the following. A maneuver is a transition from a trim trajectory given by the exponential of $\eta_1 \in \mathfrak{h}$ to a trim trajectory given by

[‡]This is possible when away from singularities, including attitudes for which either the pitch or the roll angle are equal to $\pi/2$ radians. For further details, see Chapter 5

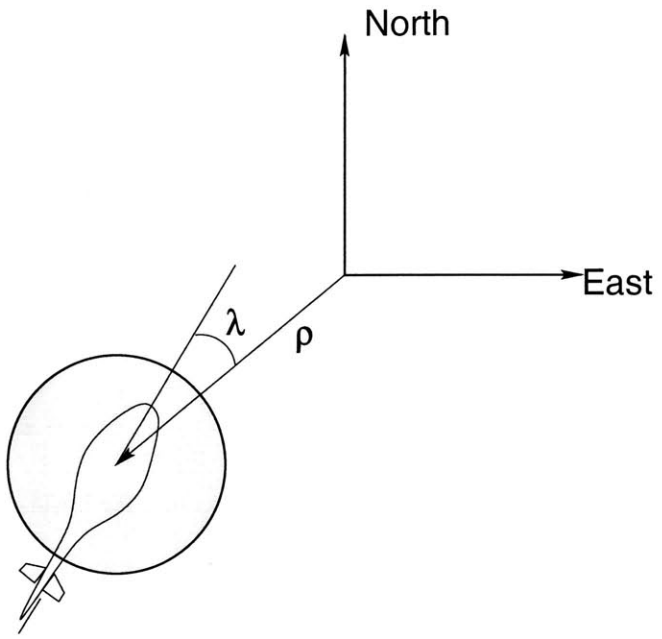

Figure 3-10: Example geometry.

the exponential of $\eta_2 \in \mathfrak{h}$. We can constrain the generated maneuver to be a piecewise-polynomial function (spline) connecting η_1 and η_2 in a smooth way, in the shortest possible time (while staying within the boundaries of the flight envelope, and the actuator saturation limits). Since the relative degree of the helicopter is four, in order to ensure continuity in the control input, we need to impose four constraints in the boundary conditions at the start of the maneuver, and four more at the end of the maneuver. Notice that the evolution of $\eta(t)$ is described by a spline, while in general the shape of the trajectory will be a more complicated curve, with derivative $\dot{h}(t) = h(t)\eta(t)$. The search for the minimum-time spline corresponding to each transition can be done via a simple bisection method.

In this example, we will consider an obstacle-free environment, and 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. 3-10).

In the example, the design trim trajectories collection is defined by:

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

Reference maneuvers are computed for transition between all trim trajectories.

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. The evaluation of the optimal cost is carried out off-line (see Fig. 3-11), and requires computation times of the order of one hour; the storage of the data requires about 1 MB of memory. The evaluation of the optimal control, that is minimization of J defined in Eq. 3.9, to be done in real-time,

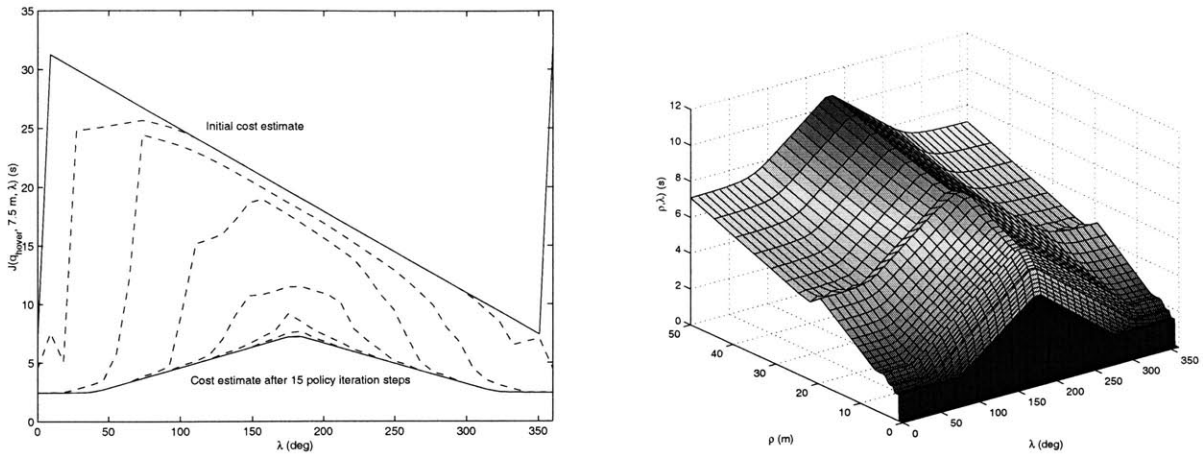

Figure 3-11: Value iteration results and optimal cost function for initial conditions at hover.

requires only a few hundredths of a second on a 300 MHz Pentium 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. 3-12. In these figures, the height of the stems represents the velocity of the vehicle; moreover, solid lines and circle symbols indicate transitions.

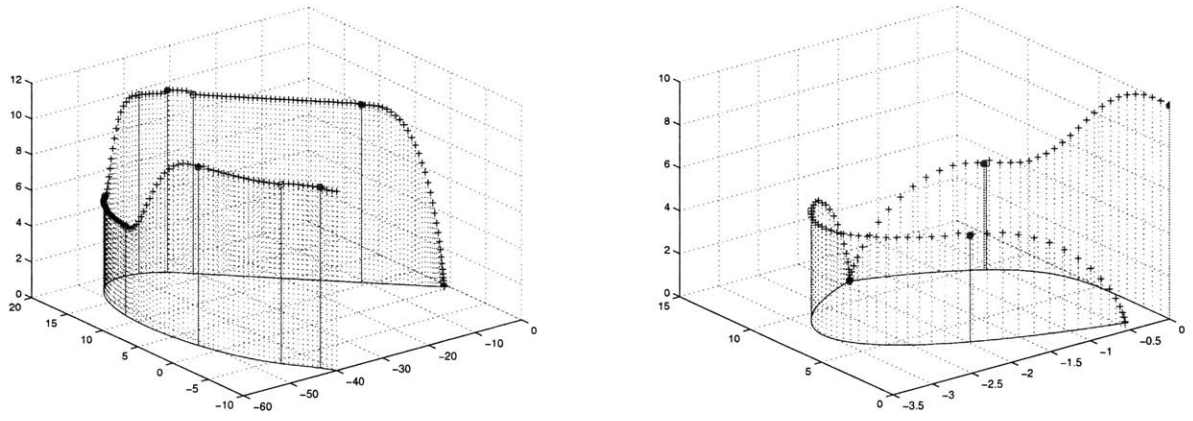

Figure 3-12: Simulated trajectory and velocity profile, starting from a high speed turn away from the target (left), and from high speed flight over the target. On the horizontal axes are reported the East and North position, in meters, and on the vertical axis is reported the magnitude of the velocity, in m/s.

Index $p \in Q_M$	Pre(p)	Next(p)	Maneuver Duration $T[s]$	Maneuver displacement $\Delta\psi[deg]$
10	0	1	4.5	44.2
11	0	2	7.5	166.8
12	0	3	16.5	794
13	0	4	4	-40.72
14	0	5	7.5	-172.9
15	0	6	16.5	-793.6
16	1	0	4	37.22
17	1	2	5	143.5
18	1	3	14.5	830.4
19	1	4	6	-12.39
20	1	5	10.5	-147.6
21	1	6	16.5	-662.7
22	2	0	6	110.5
23	2	1	4	118.8
24	2	3	10.5	644.2
25	2	4	8	51.1
26	2	5	9.5	9.222
27	2	6	18	-572.1
28	3	0	9.5	381.7
29	3	1	7.5	324.3
30	3	2	4.5	257.3
31	3	4	12	320
32	3	5	15	233.4
33	3	6	22	-331
34	4	0	3.5	-30.46
35	4	1	6	-5.828
36	4	2	10	145.3
37	4	3	19	806.9
38	4	5	5	-163.9
39	4	6	11.5	-638.3
40	5	0	5.5	-115
41	5	1	9	-83.41
42	5	2	10.5	34.59
43	5	3	21	710.2
44	5	4	3.5	-107.8
45	5	6	10.5	-665.8
46	6	0	10	-386.1
47	6	1	10	-453.2
48	6	2	14.5	-238.8
49	6	3	26	523.7
50	6	4	8	-401.2
51	6	5	6	-367.1

Table 3.2: Maneuver data for the 3-DOF helicopter.

Chapter 4

The Robust Maneuver Automaton

In this chapter we present extensions to the nominal Maneuver Automaton introduced in Chapter 3, aimed at ensuring that the trajectories generated by the Automaton are consistent with the system's dynamics, even in the presence of disturbances and unmodeled dynamics in the system and in the environment. For this purpose, we first introduce the notion of invariant tracking and stabilization, aimed at designing controllers in such a way that the symmetry of the system's dynamics is preserved under closed loop control. Second, we characterize the robustness properties of the trajectory primitives in the Maneuver Automaton, under the action of the invariant controllers. Finally, we introduce the Robust Maneuver Automaton, and give conditions for its consistency.

4.1 Introduction

In the preceding chapter we introduced the Maneuver Automaton as a new tool to implement the dynamics of a system, which enables the development of efficient motion planning algorithms. However, the discussion of the Maneuver Automaton was limited to the case in which the dynamics of the system are completely known, and the initial conditions correspond exactly to one of the trajectory primitives from which the Automaton is built. In real applications, it is generally not possible to accurately model all of the forces and torques which act on the system, and neither it is possible to set exactly the initial conditions. Nonetheless, the effect of the disturbances on the system, as well as the effect of deviations from the nominal initial conditions, must be accounted for: We will therefore need to examine the behavior of the system at non-nominal conditions, and make sure that the resulting system trajectories stay 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 $\bar{u}(t)$ stored in the Maneuver Automaton along with the reference state trajectory. Since the development of the Maneuver Automaton framework was based on the exploitation of the symmetries in the system's dynamics, it is natural to require the same invariance in the closed-loop system (that is, the vehicle subject to a feedback control law).

From the modeling point of view, we can represent the effect of external disturbances, unmodeled dynamics, and modeling errors by augmenting the control input vector u with a disturbance input vector w, which is not available to the designer. Hence, in this chapter, we will consider $n_w > 0$ in the system's equations.

4.2 Invariant Tracking and Stabilization

In the definition of the trajectory primitives, we relied on the existence of symmetries in the system's dynamics, under control signals assigned *a priori*. If the control inputs are computed as a function of the state (feedback control), the existence of symmetries is no longer ensured. On the other hand, it is possible to design the feedback control law in such a way that symmetries are preserved.

Given the initial conditions $x_0 \in \mathcal{X}$, and a control and disturbance input signal $(\bar{u}, 0)$, the nominal evolution of the system is described by $x_{ref}(t) = \phi_{(\bar{u},0)}(t, x_0)$. If, however, the disturbance input is not zero, in general the evolution of the system will be such that $x(t) = \phi_{(\bar{u},w)}(x_0, t) \neq x_{ref}(t)$. The same can be said even in the case in which the disturbance inputs w are identically zero, but the initial conditions are different from x_0.

To reject the adverse effect of the disturbance input or of non-nominal initial conditions, it is possible to add to the open-loop control signal $\bar{u}$ a (static) feedback control law $\kappa : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{U}$ which, measuring the "error" between $x(t)$ and $x_{ref}(t)$, and relying on the knowledge of the open-loop control $\bar{u}(t)$ provides a suitable control action $u(t) = \kappa(x(t), x_{ref}(t), \bar{u}(t))$, aimed at reducing the future "errors".

Assuming that the mechanical control system $\mathcal{S}$ has a symmetry group H, the question that we are interested in addresssing at this point is the following: does the closed loop system obtained by setting $u = \kappa(x, x_{ref}, \bar{u})$ maintain the same symmetry group? Since the Maneuver Automaton framework hinges on the existence of symmetries, maintaining the invariance properties of the open-loop dynamics under feedback control is convenient in addressing robustness issues of the Automaton. The problem of invariant tracking and stabilization was introduced by Rouchon and Rudolph in [113], and is based on the design of a control law such that the dynamics of an appropriately defined invariant state error are made asymptotically stable.

We can define invariance for the feedback control law in much the same way as we did for the system dynamics:

Definition 4.1 (Invariant static state feedback) *A static state feedback $\kappa : \mathcal{X} \times \mathcal{X} \times \mathcal{U} \rightarrow \mathcal{U}$ is said invariant with respect to the group H if:*

$$\kappa(\Psi_h(x), \Psi_h(x_{ref}), \bar{u}) = \kappa(x, x_{ref}, \bar{u}) \; \forall x, x_{ref} \in \mathcal{X}, h \in H, \bar{u} \in \mathcal{U}. \tag{4.1}$$

It is evident from our formulation of the system's dynamics that if the feedback control law κ is an invariant static state feedback, the symmetry properties of the open-loop system will be preserved in the closed loop (assuming no disturbances). Now the question remains of how to design an invariant state feedback. One possible way to do this is based on the construction of an invariant state error; if the control law is designed in such a way that it is a function only of an invariant error (and of the open-loop control input), it can be readily seen that it is itself invariant.

Definition 4.2 (Invariant state error) *A local (resp. global)* invariant state error *is a smooth map $e : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^{2n}$ such that, for all $x_1, x_2 \in \mathcal{X}$:*

- *$e(x_1, \cdot)$ and $e(\cdot, x_2)$ are local (resp. global) diffeomorphisms on $\mathcal{X}$*
- *$e(x_1, x_2) = 0$ if and only if $x_1 = x_2$*
- *$e(x_1, x_2) = e\left(\Psi_h(x_1), \Psi_h(x_2)\right)$*

For a mechanical system on a Lie group, a possible choice for an invariant state error is given by

$$e(x, x_{ref}) = \chi\left(g_e, \xi - \mathrm{Ad}_{g_e^{-1}}\,\xi_{ref}\right) \tag{4.2}$$

where $g_e := g_{ref}^{-1} g$ and χ is is a coordinate chart for the state manifold $(G, \mathfrak{g})$. The error is hence determined by g_e, that is the relative motion from the desired frame to the body frame, and the quantity $\xi - \mathrm{Ad}_{g_e^{-1}}\,\xi_{ref}$, which represents the difference between the velocity and the desired velocity, both expressed in body frame. Other possible choices of invariant errors for a system on a Lie group are discussed in [22]. Additional examples of invariant error functions for other kinds of systems (including differentially flat systems) are presented in [113].

Under the action of an invariant feedback control, the dynamics of an invariant state error are also invariant. Notice that this means that the state error dynamics are time invariant on a trim trajectory [53]. An invariant tracking controller is then a controller that makes the error dynamics asymptotically stable. One possible approach for tracking control design is based on the definition of a Control Lyapunov function $V(e)$ on the state error, and the computation of a control input that makes its derivative along trajectories $L_{\dot{e}}V$ negative definite [5, 125, 37].

4.3 Robustness characterization of trajectory primitives

The trajectories generated by the nominal Maneuver Automaton, as introduced in Chapter 3, can be regarded as reference trajectories which have to be tracked by the actual system. In order to decouple tracking/stabilization and motion planning concerns, we want to ensure that, for each trajectory primitive in the Maneuver Automaton, a feedback control law is available that keeps the system "close" to the nominal trajectory. In this section we will look at the characterization of such "closeness" for the two different classes of trajectory primitives on which the Maneuver Automaton is built.

4.3.1 Relative Equilibria

Relative equilibria are by definition trajectory primitives which characterized by an infinite time horizon: it is in principle possible that the system be required to stay "close" to a trim trajectory for an indefinite amount of time. As a consequence, the robustness characteristics of equilibrium points can be expressed in terms of invariant sets.

Recall that, on a relative equilibrium (identified by the index $q \in Q_T$), the nominal trajectory is described by Eq. (3.1):

$$x_{ref}(t) = \Psi(h_0 \exp(\bar{\eta}_q t), \bar{x}_q) = \Psi_{h(t)}(\bar{x}_q) \tag{4.3}$$

where $h(t) = h_0 \exp(\bar{\eta}_q t)$ is the nominal position on the symmetry group.

Given an invariant state error e, the deviation from the nominal trajectory is (dropping the explicit dependence on time of x,x_{ref}, and h):

$$e\left(x, x_{ref}\right) = e\left(x, \Psi_h(\bar{x}_q)\right) = e\left(\Psi_h^{-1}(x), \bar{x}_q\right) = e_q\left(\Psi_h^{-1}(x)\right). \tag{4.4}$$

Under the action of a control law κ, the evolution of the state error $\phi^e_{(\kappa,w)}$ can be obtained from the actual and nominal state flows:

$$\phi^e_{(\kappa,w)}(t, e_0) = e(\phi_{(\kappa,w)}(t, h_0\bar{x}_q), \Psi_{h_0 \exp \eta_q t}\bar{x}_q), \tag{4.5}$$

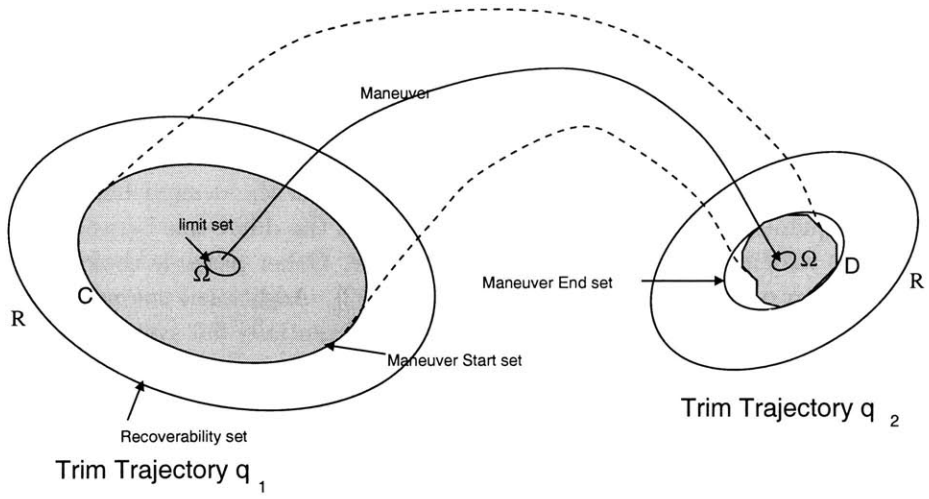

Figure 4-1: Invariant set definitions.

with the initial state error being defined by $e_0 = e(x(0), x_{ref}(0))$.

Definition 4.3 (Invariant set) *A set $M \subseteq \mathbb{R}^n$ is said to be an invariant set if for all $e_0 \in M$, $w \in \mathcal{W}$, and $t > 0$*

$$\phi^e_{(\kappa,w)}(t, e_0) \in M$$

Invariant sets correspond to "tubes" centered on trim trajectories, with a constant "section" (and M corresponds to the section at the identity on the symmetry group). If a Lyapunov function is knwon for the closed loop system, invariant sets correspond to level sets of the Lyapunov function.

Assuming that the magnitude of the disturbance input is bounded, in general it is possible to design a feedback control law that maintains the error bounded. In particular, it is of interest to characterize what is the smallest invariant set to which it is possible to reduce the error, given enough time.

Definition 4.4 (Limit set) *The limit set of a trim trajectory q is the smallest invariant set $\bar{\omega}_q$ associated with that trajectory.*

Since the limit set can in principle have zero dimension, or require infinite time for the system to enter it, we will also consider a *practical limit set*, as an arbitrary (in general small, but finite) invariant set $\bar{\Omega}_q \supseteq \bar{\omega}_q$. The objective of the feedback control law acting on the automaton will be to maintain the state error within this practical limit set, whenever the system is executing a trim trajectory.

Definition 4.5 (Recoverability set) *The recoverability set $\bar{R}_q$ of a trim trajectory is the largest set for which there exists a finite time $\tilde{t}$ such that for all initial conditions $e_0 \in \bar{R}_q$, and for all disturbance signals $w \in \mathcal{W}$, the errors enters the (practical) limit set, that is if:*

$$\phi^e_{(\kappa,w)}(t, e_0) \in \bar{\Omega}_q, \ \forall\, t > \tilde{t}. \tag{4.6}$$

In general, the exact determination of the sets $\bar{R}_q$ 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 $R_q \subseteq \bar{R}_q$ such that for all initial conditions in 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. Level sets of (control) Lyapunov functions are particularly well suited for this purpose.

It is obviously of interest to design a control law in such a way to have a large R_q, and a small Ω_q. In the case in which the control law provides global stability, R_q will coincide with X, and in the case in which we have asymptotic stability, Ω_q can in principle be made as small as desired (in the general case, this is possible only if the disturbance signal w vanishes, or can be measured and exactly matched by the control). However, in the design of the Maneuver Automaton, it will be more appropriate to set the practical limit set to be large enough to be attained "quickly", without requiring excessive settling times.

4.3.2 Maneuvers

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 4.6 (Image of a set) *We will define the image of a set C_p under the maneuver $p \in Q_M$ the smallest set $D_p \in \mathbb{R}^{2n}$ such that for all $e_0 \in C_p$, and for all disturbance signals $w \in \mathcal{W}$ (supported on $[0, \bar{T}_p)$),*

$$\phi^e_{\kappa,w}(\bar{T}_q, e_0) \in D_p.$$

The objective of the control law is to make the ending set D_p as small as possible for a given starting set C_p. 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_p.

4.4 The Robust Maneuver Automaton

At this point we can extend the definition of the nominal Maneuver Automaton, given in the previous chapter, in order to address robustness issues.

We will call such a control architecture a Maneuver Automaton:

Definition 4.7 (Robust Maneuver Automaton) *A Robust Maneuver Automaton (RoMA) over a mechanical control system S, with symmetry group H, is described by the following objects:*

- *A finite set of indices $Q = Q_T \cup Q_M \subset \mathbb{N}$, where the subscript T relates to trim trajectories, and the subscript M relates to maneuvers.*
- *A finite set of trim trajectory parameters $(\bar{\xi}, \bar{g}, \bar{u})_q$, with $q \in Q_T$. Define $\bar{\eta}_q = \mathrm{Ad}_{\bar{g}_q} \bar{\xi}_q$.*
- *A finite set of maneuver parameters, and nominal state and control trajectories $(T, \bar{u}, \bar{\phi})_q$, with $q \in Q_M$.*
- *The maps $\mathrm{Pre} : Q_M \to Q_T$, and $\mathrm{Next} : Q_M \to Q_T$ such that $\mathrm{Pre}(q)$ and $\mathrm{Next}(q)$ give, respectively, the index of the trim trajectories from which the maneuver q starts and ends.*

- *A finite set of control laws* κ_q, $q \in Q_T \cup Q_M$.
- *A collection of invariant sets* Ω_q, R_q, $q \in Q_T$
- *A collection of maneuver starting and ending sets* C_p, D_p, $p \in Q_M$
- *A continuous controller state* $h \in H$
- *A discrete controller state* $q \in Q$.
- *A clock state* $\theta \in \mathbb{R}$, *which evolves according to* $\dot{\theta} = 1$, *and which is reset after each switch on* q.

4.4.1 Automaton Consistency

In order to decouple stability and motion planning concerns we must make sure that, given any sequence of hybrid controls (trim trajectory-maneuvers sequences), the actual trajectory of the controlled system will remain "close" to the nominal trajectory, in the presence of disturbances. We will call this property *consistency*:

Definition 4.8 (Automaton consistency) *We say that a Robust Maneuver Automaton is* consistent *if, given any initial conditions* $(q, h, x)_0$, *with initial error* $e_0 \in \Omega_{q_0}$ *and any sequence of hybrid inputs* $\bar{\upsilon}$, *the following is always true:*

$$e(x, h\bar{g}_q) \in \Omega_q.$$

In other words, a Robust Maneuver Automaton is said to be consistent if all trajectories generated by the Automaton are executable by the system, even in the presence of disturbances, up to a known error bound. In this sense, a consistent Robust Maneuver Automaton enables the implementation of robust motion planning algorithms.

In order to ensure that a Robust Maneuver Automaton is indeed consistent, we can check for the following condition:

Proposition 4.1 (Sufficient conditions for consistency) *A Robust Maneuver Automaton is* consistent *if for all* $p \in Q_M$ *the following conditions hold:*

1. $C_p \supseteq \Omega_{\mathrm{Pre}(p)}$;
2. $D_p \subseteq R_{\mathrm{Next}(p)}$;
3. $D_p \subseteq \Omega_{\mathrm{Next}(p)}$;

Notice that 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. A possible way to generate this recovery phase would be to use the controller associated with the new trim trajectory, which is known to take the error into the invariant set $\Omega_{\mathrm{Next}(p)}$ in finite time.

4.5 Examples

It is apparent at this point what the procedure for "robustifying" a Maneuver Automaton is:

1. Build a nominal Maneuver Automaton.
2. Design invariant feedback controllers for each primitive in the Automaton.
3. Compute estimates of limit and recoverability sets on trim trajectories.
4. Compute Poincaré maps for the maneuvers.
5. If consistency conditions are not met, extend maneuvers by recovery times, or possibly relax the tracking requirements by choosing larger practical limit sets.

In the following, we will give a simple example of the construction of such Robust Maneuvers Automata. A more complex example, involving the design of an invariant tracking feedback control law for a small autonomous helicopter, will be the subject of the next chapter.

4.5.1 Double Integrator

As a first, simple example, consider again the double integrator of Section 3.4.1, with constraints on maximum acceleration and velocity ($|u| \leq 1$ and $|\dot{y}| \leq 1$ in non-dimensional units). In addition, this time we will also consider the effect of a disturbance force (e.g. drag) on the system. We will assume that the magnitude of the disturbance force, and the resulting acceleration w, are bounded, and smaller than the maximum control magnitude: $|w| \leq \epsilon < 1$. The following treatment of the double integrator case, based on level sets of quadratic Lyapunov functions, can be easily extended to higher-dimensional linear systems.

Setting $x_1 = y$, and $x_2 = \dot{y}$, a state-space formulation of the dynamics of the system is the following:

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix}(u+w). \tag{4.7}$$

An invariant state error is given by:

$$e(x, x_{ref}) = (x - x_{ref}).$$

An invariant tracking controller can be derived by choosing a control cost $\rho^2 > 0$ and solving for the unique stabilizing solution of the Riccati equation:

$$A^T P + PA - \frac{1}{\rho^2} PBB^T P + I = 0,$$

with $A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$, $B = [0,\ 1]^T$, and setting

$$u = \text{sat}(u_{ref} - \frac{1}{\rho^2} B^T P e),$$

with u_{ref} corresponding to the nominal open-loop control input (i.e. $\dot{x}_{ref} = Ax_{ref} + Bu_{ref}$). Simple algebra yields:

$$P = \begin{bmatrix} \sqrt{2\rho+1} & \rho \\ \rho & \rho\sqrt{2\rho+1} \end{bmatrix}. \tag{4.8}$$

Choosing the Lyapunov function $V(e) = e^T P e$, its derivative along trajectories of the system satisfies:

$$\frac{d}{dt}V(e) = 2e^T P\dot{e} = -2e^T P(Ax + Bu + Bw - \dot{x}_{ref}).$$

Assuming that the control input u does not saturate ($|B^T Pe| < \rho^2(1 - |u_{ref}|)$), we have:

$$\frac{d}{dt}V(e) = -2e^T e + 2e^T PBw \leq -e^T e + \epsilon\|e\|_2\|PB\|_2.$$

An estimate of the limit set is obtained by choosing the smallest invariant set determined by contour levels of V. In our case, the derivative of V along trajectories of the system is guaranteed to be negative outside the ball of radius $e_\Omega = \epsilon\rho\sqrt{2(\rho+1)}$, centered at the origin. Hence, an estimate of the limit set is given by the level set $V(e) < V_\Omega := e_\omega^2 \lambda_{max}(P)$, where $\lambda_{max}(P)$ is the largest eigenvalue of P; the corresponding recoverability set is the whole space.

The magnitude of the maximum velocity error is given by $\sqrt{P_{11}V_\Omega/\det(P)}$. Hence, in order to ensure envelope protection, when selecting trim trajectories, the maximum magnitude of the nominal velocity must be chosen so that $v_{max} \leq 1 - \sqrt{P_{11}V_\Omega/\det(P)}$. The position error is guaranteed to be bounded by $\sqrt{P_{22}V_\Omega/\det(P)}$, which determines the size of the "tube" around the trajectory in the physical space.

Moreover, the maximum magnitude of the feedback control input is given by $\frac{1}{\rho^2}\sqrt{P_{22}V_\Omega}$. Hence, if maneuvers are designed in such a way that the maximum nominal acceleration is $u_{max} \leq 1 - \frac{1}{\rho^2}\sqrt{P_{22}V_\Omega}$ then the invariance of level sets $V(e) < V_\Omega$ is preserved under maneuvers as well (in the sense that if the maneuver start sets C_p are contained in level sets $V(e) < V_\Omega$, so will be the corresponding maneuver end sets D_p). This corresponds to limiting the "aggressiveness" of nominal maneuvers to maintain the control authority needed to keep the state error small enough to ensure the consistency of the automaton. An alternative approach would be to design nominal maneuvers using higher acceleration levels, and "specialized" tracking controllers for maneuvers; if needed, recovery phases at the end of each nominal maneuver can be introduced.

As an example, assume $\epsilon = 0.1$, and choose $\rho = 0.5$. To guarantee "flight envelope" protection, the maximum nominal velocity magnitude must be chosen as $v_{max} \leq 0.8462$, and the maximum nominal acceleration magnitude as $u_{max} \leq 0.6232$ (if we want to maintain the same tracking controller for all trajectory primitives). The magnitude of the position error will be bounded by $e_{1,max} = 0.1088$. An example of the actual trajectory of a system governed by a Robust Maneuver Automaton, subject to a worst-case disturbance, is given in Figure 4-2, together with its nominal trajectory.

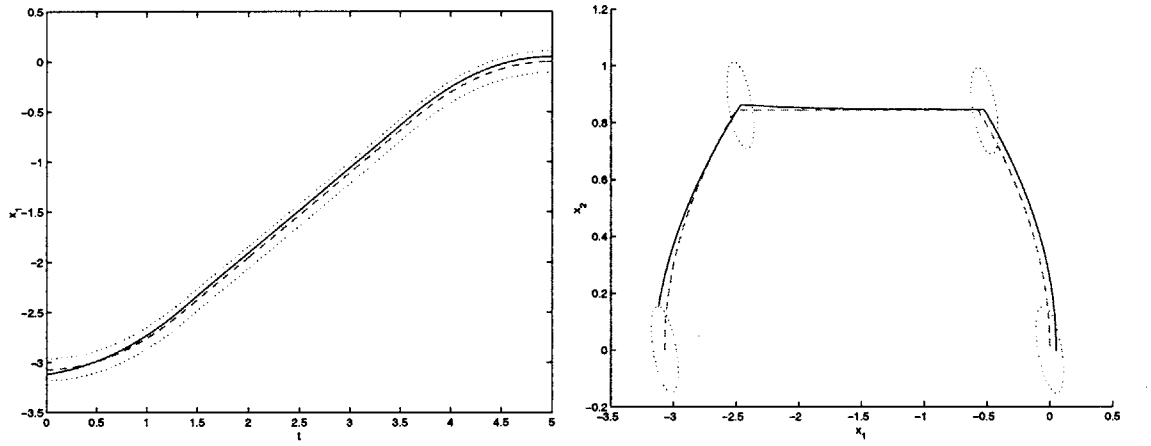

Figure 4-2: Simulated trajectory for a robust maneuver automaton on a double integrator with worst-case acceleration disturbances.

Chapter 5

Invariant tracking control design for autonomous helicopters

In this chapter we present a control design technique for a class of nonlinear, underactuated mechanical systems, based on a backstepping procedure. This class includes an approximation of small helicopter dynamics. The need to avoid artificial singularities due to the attitude representation is the main driver behind the control design presented in this chapter: to achieve this goal, we will operate directly in the configuration manifold of the vehicle. The control design provides asymptotic tracking for an approximate model of small helicopters, and bounded tracking when more complete models are considered, and in the presence of bounded disturbances. Since the control design method is based on the construction of a Lyapunov function on an invariant state error, it is relatively easy to estimate invariant sets and Poincaré maps for the closed-loop system. The computation of these estimates can be used to check the nesting conditions (4.1) for consistency of the Robust Maneuver Automaton.

5.1 Background

In the recent past, the design and implementation of control algorithms for autonomous helicopters has been the object of a relevant body of research, due to an identified need for maneuverable autonomous aerial vehicles, for both military and civil applications. While slower and less fuel-efficient than airplanes, helicopters are capable of vertical take off and landing, hover, and in general are more maneuverable in tight spaces than airplanes. As a consequence, helicopters are one of the best platforms for operations in urban or otherwise cluttered environments. However, in many respects the dynamics of a helicopter are more complicated than the dynamics of a fixed wing aircraft: A helicopter is inherently unstable at hover, and the flight characteristics change dramatically over the entire flight envelope.

In recent papers, feedback linearization techniques have been applied to helicopter models. 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 [42, 62] The feedback linearization approach suffers from the fact that, since the attitude is usually parameterized using Euler angles, singularities arise when performing some maneuvers, such as loops, barrel rolls, split-S's and

others. 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 models 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 to the attitude representation is the main driver behind the control design presented in this chapter: To achieve this goal, we will operate directly in the configuration manifold of the helicopter. The "tracking on manifolds" problem was presented and solved in [22] for fully actuated mechanical systems: In this chapter we present an extension, for achieving asymptotic (locally exponential) tracking of trajectories for a particular class of underactuated mechanical systems. An approximate model of helicopter dynamics can be shown to be in this class: The approximation that will be used in the following is the same one that leads to feedback linearizability, or differential flatness of the model. However, the method presented here can deal without any modification with more accurate models, including for example simple aerodynamic forces.

The control design will be based on a non-trivial extension of backstepping ideas [64] to dynamic systems on manifolds. In its basic form the backstepping procedure is carried out on a chain of integrators (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 [39] for control of surface vessels: Our problem is more difficult since we need to control the rigid body motion in the three-dimensional space, as opposed to the plane. A very similar, yet independently developed control design method was presented in [83], where a backstepping procedure was implemented to control 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"[83]. 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).

In the rest of the chapter, we will first present the details of the control design method. The fundamental idea behind the method is based on the decomposition of the dynamics into a translational block and a rotational block. The translational dynamics, even though underactuated, can be modelled as a controllable, linear system. This linear system is driven by the attitude dynamics, which is a fully actuated nonlinear system, for which it is relatively easy to track any given trajectory. The idea is to steer the attitude of the system in such a way that the translational dynamics are stabilized. When doing this, we must of course ensure stability of the overall closed-loop system. The design process can be split into three parts: Control of the translational dynamics, control of the attitude, and the backstepping procedure to construct a stabilizing (or tracking) controller for the overall system. Since the controller design procedure will be carried out on an approximate model, in the following section we will assess the impact of the neglected dynamics. At that point we will be able to derive invariant sets for the closed-loop error dynamics, and check consistency of an automaton built through the methodology presented in Section 3.4.3, augmented with an invariant control law as explained in Chapter 4. Finally we will present and discuss some simulation examples.

5.2 Control Design

A model of the dynamics of a small helicopter was introduced in Chapter 2. The objective of the feedback control law will be to track a smooth, feasible reference trajectory $(g_{ref}(t), \hat{\xi}_{ref}(t))$ (at this point we are not limiting ourselves to trajectories generated by a Maneuver Automaton). We will start by designing a controller for the approximate system obtained by setting $\mathcal{E} = 0$ in (2.10), and assuming that the non-conservative drag forces $b(g, \xi)$ are attitude-independent (e.g. in the inertial frame the drag can be expressed as a function of $\dot{p}$ only, i.e. $-\tilde{b}(\dot{p})$). The assumption that $\epsilon_i = 0$ means that the control inputs correspond to three independent moments and a force that acts in a body-fixed direction (the rotor shaft*). We are neglecting the fact that in order to generate moments, force components acting in directions normal to the rotor shaft are required; however, the magnitude of these forces is small when compared to the main thrust (see also Figure 2-2).

This approximation simplifies the control design considerably: It can be shown that the resulting approximate system is differentially flat, and hence feedback linearizable [88, 137, 62]. 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.

5.2.1 Translational dynamics

If we consider as translational coordinates the position and velocity in the inertial frame $(p, \dot{p}) = (p, Rv)$, the translational dynamics (2.9) can be rewritten as three double integrators in parallel, subject to gravity and non-conservative, attitude-independent forces, and driven by the force input $\alpha(R, u_4) := Rf(u)$:

$$\ddot{p} = m\tilde{g} - \tilde{b}(\dot{p}) + \alpha(R, u_4).$$

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

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

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. In other words, if we were able to control precisely the attitude, it would be trivial to stabilize the translational dynamics. The above can be easily extended for tracking of a reference trajectory $p_{ref}(t)$, by adding appropriate feed-forward terms. For simplicity, we can use a quadratic Lyapunov function and a proportional-derivative (PD) form for the translational dynamics control law:

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

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. The translational controller thus derived is clearly invariant with respect to translations; additionally, it is invariant with respect to rotation about a vertical axis if $K_p = \mathrm{diag}(k_{p,h}, k_{p,h}, k_{p,v})$, and $K_{\dot{p}} = \mathrm{diag}(k_{\dot{p},h}, k_{\dot{p},h}, k_{\dot{p},v})$, where the subscripts h and v distinguish horizontal and vertical gains.

If the function $\alpha(R, u_4)$ were invertible, we would be able to use the attitude R as a control input to the translational block. This is not the case: Imposing the condition

*Alternatively, one could consider a slightly slanted body-fixed direction, to take into account the offset due to the tail rotor side force at hover.

$\alpha(R, u_4) = v$ determines R up to an arbitrary rotation about the direction v. However, we can select the desired attitude R_d as the "closest" (in the sense explained below) element of $SO(3)$ to the 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 $\delta R := R_2^{-1} R1$ that is still an element of $SO(3)$ (this is an invariant configuration error). All the elements of $SO(3)$ (the group of orthogonal 3×3 matrices with determinant $+1$, or rotation matrices) 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 attitude error δR, that is the distance between the rotations R_1 and R_2, we can consider the following function [59]:

$$\Theta(\delta R) := 1 - \cos(\tilde{\theta}) = 2 \sin^2 \frac{\tilde{\theta}}{2} = \frac{1}{2} \mathrm{Tr}(I - \delta R). \tag{5.1}$$

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

$$\begin{aligned} (R_d, u_{4d}) \quad = \quad & \underset{(R,u) \in SO(3) \times \mathbb{R}}{\arg\min} \quad \Theta(R_{ref}^{-1} R) \\ & s.t. \quad \alpha(R, u) = \bar{\alpha}(p, \dot{p}). \end{aligned} \tag{5.2}$$

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 translational dynamics are independent from 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 (5.2), 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 minimization problem (5.2):

$$(R_d, u_{4d}) = \begin{cases} \left(\mathrm{Rot}(-r, \sin^{-1} ||r||_2) R_{ref}, \ ||\bar{\alpha}(p, \dot{p})||_2\right), \\ \qquad\qquad \text{if } R_{ref} e_3 \cdot \bar{\alpha}(p, \dot{p}) \leq 0, \\ \left(\mathrm{Rot}(r, \sin^{-1} ||r||_2) R_{ref}, \ -||\bar{\alpha}(p, \dot{p})||_2\right), \\ \qquad\qquad \text{if } R_{ref} e_3 \cdot \bar{\alpha}(p, \dot{p}) > 0 \end{cases} \tag{5.3}$$

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:

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

5.2.2 Attitude dynamics and backstepping control design

Once we have the desired attitude, using ideas derived from the backstepping concept [64], we want to track R_d to stabilize the overall system. However, before we can go on with the

[†]Based on the fact that the smallest amplitude rotation to superimpose one unit vector onto another is a rotation about the normal to the plane determined by the two unit vectors.

control design, we must look at the rate of change of the objects introduced in the previous section. First of all, we have $\hat{\omega}_{ref} = R_{ref}^T \dot{R}_{ref}$; accordingly, we can define $\hat{\omega}_d := R_d^T \dot{R}_d$. Furthermore, the following equalities hold:

$$\begin{aligned} \frac{d}{dt}\Theta(R_d^T R) &= -\frac{1}{2}\,\mathrm{Tr}(\dot{R}_d^T R + R_d^T \dot{R}) = -\frac{1}{2}\,\mathrm{Tr}(\hat{\omega}_d^T \dot{R}_d^T R + \dot{R}_d^T R \hat{\omega}) \\ &= -\frac{1}{2}\,\mathrm{Tr}\left(R_d^T R(\hat{\omega} - \hat{\omega}_d)\right) = \mathrm{Skew}(R_d^T R)^\vee \cdot (\omega - \omega_d) \\ &= \sin\tilde{\theta}\tilde{r}_d \cdot (\omega - \omega_d) \end{aligned} \tag{5.4}$$

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 $\tilde{r}_d, \tilde{\theta}$ are respectively the fixed axis and rotation angle of the attitude error $R_d^T R$, obtained by

$$\cos\tilde{\theta} = \frac{\mathrm{Tr}(R_d^T R) - 1}{2}; \quad \sin\tilde{\theta}\tilde{r}_d = \mathrm{Skew}(R_d^T R)^\vee.$$

We can finally rewrite Eq.(5.4) as

$$\frac{d}{dt}\Theta(R_d^T R) = \nabla\Theta \cdot (\omega - \omega_d),$$

having defined $\nabla\Theta := \sin\tilde{\theta}\tilde{r}_d = \mathrm{Skew}(R_d^T R)^\vee$. The quantity $\omega - \omega_d$ is an invariant attitude rate error (as defined in Chapter 4).

Now we are ready to state the following result :

Theorem 5.1 (Tracking for the approx. 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. (2.10), with $\epsilon_i = 0, i = 1 \ldots 4$, there exists a static, invariant feedback control law under which the system state $x(t)$ tracks the reference trajectory $x_{ref}(t)$. Tracking convergence is almost globally asymptotic, and locally exponential.*

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 penalize the attitude configuration and velocity errors. Such a candidate Lyapunov function is the following:

$$\begin{aligned} V = V_p(p, \dot{p}) &+ k_\theta \Theta(R_d^T R) \\ &+ \frac{1}{2}\left(||\eta||_2^2 + k_u||u_4 - u_{4d}||_2^2 + ||\zeta||_2^2\right) \end{aligned} \tag{5.5}$$

where

$$\begin{aligned} \eta &:= \omega - \omega_d - \frac{u_{4d}}{k_\theta \cos(\tilde{\theta}/2)} R_d^T \left(\tilde{t} \times \nabla_{\dot{p}} V_p\right) \\ \zeta &:= \dot{u}_4 - \dot{u}_{4d} - \frac{1}{k_u}\nabla_{\dot{p}} V_p \cdot R e_3, \end{aligned}$$

and

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

Computing the time derivative of V, with the definition of R_d and u_{4d} given in (5.2), we get

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

We can make the above negative semidefinite by imposing:

$$\dot{\eta} = -k_\eta \eta - k_\theta \nabla\Theta \tag{5.6}$$
$$\dot{\zeta} = -k_\zeta \zeta - k_u(u_4 - u_{4d}) \tag{5.7}$$

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

$$\begin{aligned}
&\nabla_{\dot{p}} V_p \cdot (u_{4d} R_d e_3 - u_4 R e_3) = \\
&\quad = -\nabla_{\dot{p}} V_p \cdot \left[2u_{4d} \sin\frac{\tilde{\theta}}{2} (\tilde{r} \times \tilde{t}) + (u_4 - u_{4d}) R e_3 \right] \\
&\quad = k_\theta \nabla\Theta \cdot (\eta - \omega + \omega_d) + k_u (u_4 - u_{4d})(\zeta - \dot{u}_4 + \dot{u}_{4d}),
\end{aligned}$$

we get

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

The time derivative along system trajectory of the Lyapunov function V is hence negative-semidefinite, for all initial conditions such that $V(x_0) < k_\theta$ (which guarantees that $\tilde{\theta} < \pi/2$). Asymptotic stability can be inferred from LaSalle's principle. To prove local exponential stability, augment the Lyapunov function (5.5) with a cross term

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

where χ is a small positive constant. The time derivative of the cross term, under the control law (5.6) can be computed as

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

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

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

and

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

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 $V_{total} := V + V_{cross}$ is negative definite, and it is possible to find a $\lambda > 0$ such that $\dot{V}_{total} < -\lambda V_{total}$, which proves local exponential stability. The explicit expression for the control torques $\mathbf{u} = [u_1, u_2, u_3]^T$, and the control force second derivative $\ddot{u}_4$ are given by

$$\ddot{u}_4 = \ddot{u}_{4d} - k_\zeta \zeta - k_u (u_4 - u_{4d}) + \tfrac{1}{k_u} \frac{d}{dt} \left(\nabla_{\dot{p}} V_p \cdot R e_3 \right)$$

$$\begin{aligned}
\mathbf{u} = \; &\omega \times \mathbb{J}\omega - \beta(g, \xi) + \mathbb{J} \Big[-k_\eta \eta - k_\theta \nabla\Theta + \frac{d\omega_d}{dt} + \\
&+ \frac{d}{dt} \left(\frac{u_{4d}}{k_\theta \cos(\tilde{\theta}/2)} R_d^T \left(\tilde{t} \times \nabla_{\dot{p}} V_p \right) \right) \Big]
\end{aligned}$$

■

To the author's 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 [22], the new control law 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 (5.2.2) is the absence of artificial singularities, deriving from attitude parameterizations, like Euler angles. The elimination of geometric singularities has been accomplished through an over-parameterization of the outputs: We need to fully specify the reference attitude that we want to track. In order to achieve asymptotic tracking, we need the full trajectory to be feasible, and the reference attitude has to satisfy the constraints represented by the system dynamics. On the other hand 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. (5.1)) 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, notice that imposing (5.3) 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.

5.2.3 Tracking for the actual model

So far, we designed a controller to achieve asymptotic tracking of a reference trajectory for the approximate system. The dynamic terms neglected in the approximation appear as perturbations in the nominal model. A first question that arises is about the controller's behavior on the actual system, and in the presence of bounded external forces $||F_e||_2 \leq \Delta_e$, like for example those arising from uncertainties in the aerodynamics, or from wind gusts.

We want to analyze the robustness properties of the control law given in Eq. (5.2.2), in order to construct a consistent automaton, as defined in Chapter 4. Assume that the reference trajectory $x_{ref}(t) = (g_{ref}(t), \xi_{ref}(t))$ is feasible for a rigid body under the forces in (2.10). Note that if we replace the nominal thrust direction in body axes e_3 with the actual thrust direction as obtained from Eq. (2.10), the control laws (5.2.2) will give exact tracking for initial conditions on the reference trajectory, and no external disturbances. Since the unmodeled forces F_u are a function of the control, given in Eq. (2.10), that is a smooth function of the states and the reference trajectory, we have that, in a compact set $R := x \in X, V(x, x_{ref}) \leq \bar{V}$, we can characterize the effect of the neglected coupling as:

$$||F_u||_2 \leq \Delta_u + \epsilon||\delta||_2$$

where δ is the state error vector $\delta = [p - p_{ref}, \dot{p} - \dot{p}_{ref}, \nabla\Theta, \eta, u_4 - u_{4d}, \zeta]^T$. Also, it will be possible to write $W(P, P_{ref}) < \alpha_w||P||_2^2$. We have the following:

Theorem 5.2 (Bounded tracking for the full model) *Given a reference trajectory $x_{ref}(t) = (g_{ref}(t), \xi_{ref}(t))$ for a rigid body under the action of the forces in Eq. (2.10), there exist sufficiently small $\epsilon, \Delta = \Delta_u + \Delta_e$ such the control law defined in Section (5.2.2) 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

$$\begin{aligned}
\dot{V}_{total} &\leq -\lambda V_{total} + \nabla_{\dot{p}} V (F_e + F_u) \leq \\
&\leq -\lambda V_{total} + \beta ||\delta||_2 \, (\Delta + \epsilon ||\delta||_2) \leq \\
&\leq -(\lambda - \beta\epsilon\mu) V_{total} + \beta\Delta\sqrt{\mu V_{total}}
\end{aligned}$$

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

$$\Omega = \left\{ x \in \mathcal{X} \mid V_{total}(x, x_{ref}) < \mu \left(\frac{\beta\Delta}{\lambda - \beta\epsilon\mu} \right)^2 \right\}. \tag{5.8}$$

If $\epsilon < \lambda/(\beta\mu)$, $\dot{V}_{total}$ is negative semi-definite in the set $R \setminus \Omega$. Note that this, to be of any significance, requires that Δ be small enough that $\Omega \subset R$, 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 5.3).

The set defined in (5.8) is an invariant set for all feasible trajectories, as long as the controls do not saturate (in the design of the invariant tracking control law, we did not take control saturation into account). As such, a Maneuver Automaton built on the approximate system of Section 2.3.3, including for example the Automaton in Section 3.4.3, can be made a consistent Robust Automaton by augmenting all of its trajectory primitives by the control law (5.2.2), and the invariant sets (5.8).

5.3 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 parameterization.

The first example 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. 5-1, 5-2 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 noticeable differences between the behavior of the approximate model and of the actual model, we had to multiply the ϵ_i values for a typical model helicopter by a factor of 10. 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 (up to a rotation of 180 degrees). However, we notice the typical "undershoot" of non-minimum phase systems in the response of the actual system in the inverted flight case . 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 $\ddot{\phi} = -k \sin\phi$. We notice the non-minimum phase

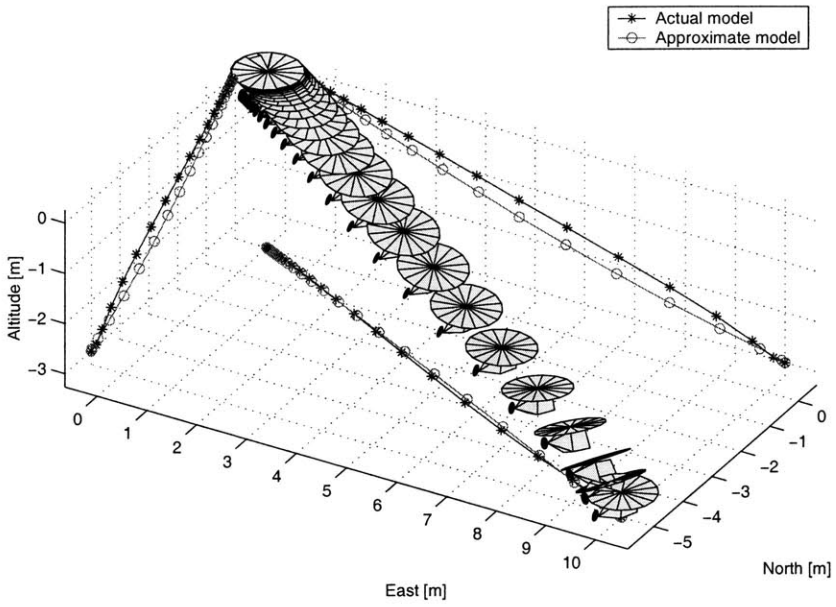

Figure 5-1: Point stabilization example.

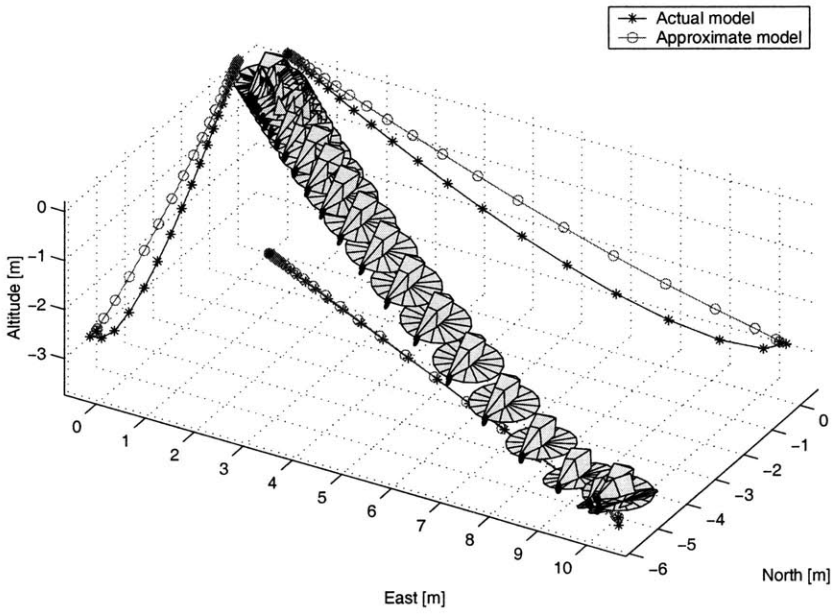

Figure 5-2: Point stabilization example — inverted flight.

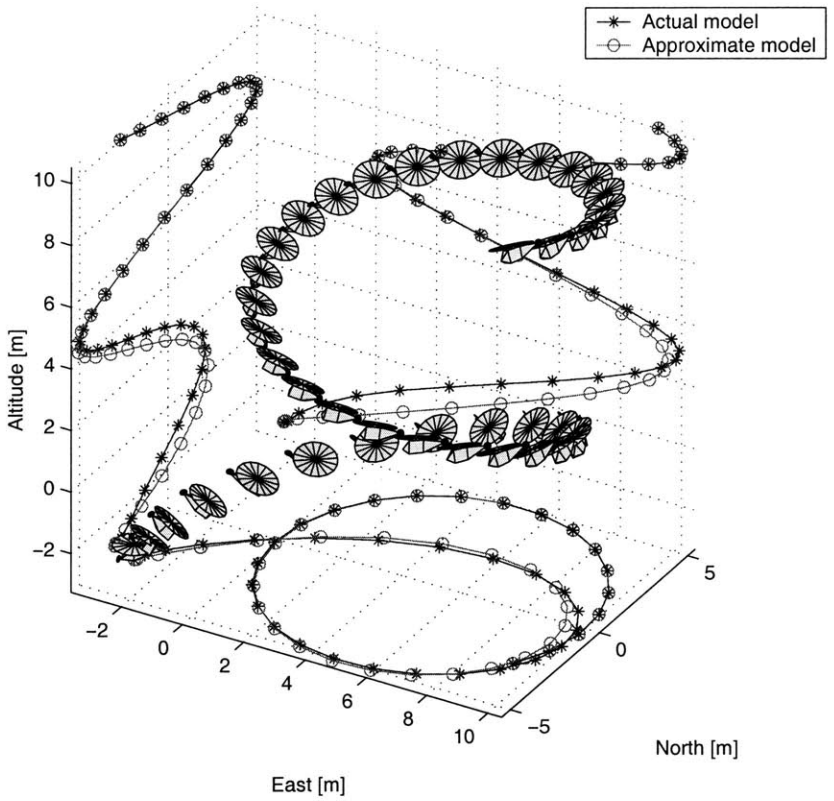

Figure 5-3: Trim trajectory tracking.

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. 5-3, is about tracking 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 [41, 53] could be profitably used in this case, but we will not pursue this direction here.

As a fourth example, we will consider a loop, that is a maneuver that is "difficult" to execute if the tracking control law is formulated in terms of Euler angles, or other singular parameters. To handle the singularity, one could switch coordinate charts, but this is likely to result in discontinuities in the control input or other undesirable behaviors, not to mention the increase in the complexity of the control software. Another option would be to relax the tracking problem to perform attitude tracking only during the maneuver, and switch back to a position tracking control law at the end of the maneuver. However, this could result in considerable deviations from the nominal path. Our control law, on the other hand, is capable of tracking the loop maneuver in a very smooth fashion, even when starting with a considerable initial error, as can be seen in Figure 5-4. Notice that the helicopter tracks both the reference position and the reference attitude (i.e. it actually performs the loop).

As a last example, in Fig. 5-5 we consider tracking a trajectory that performs a transition to inverted flight. This maneuver, in the case in which continuous controls are required,

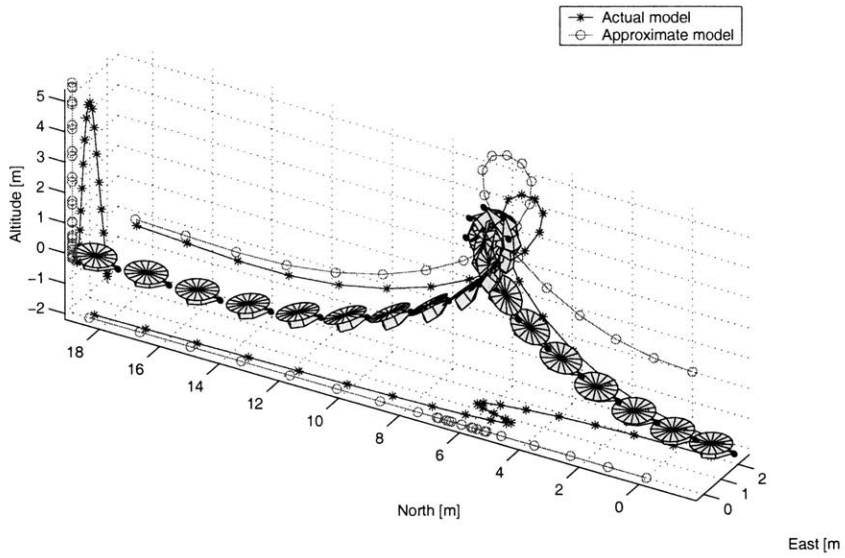

Figure 5-4: Loop tracking example.

must 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}. However, the increment ΔV between across the singularity can be made arbitrarily small, for example by reducing T_{min}, at the expense of a larger required control authority. As a matter of fact, the rotor thrust can be changed very quickly on helicopters , since the kinetic energy stored in the rotor can be used to provide very fast commanded thrust responses. 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).

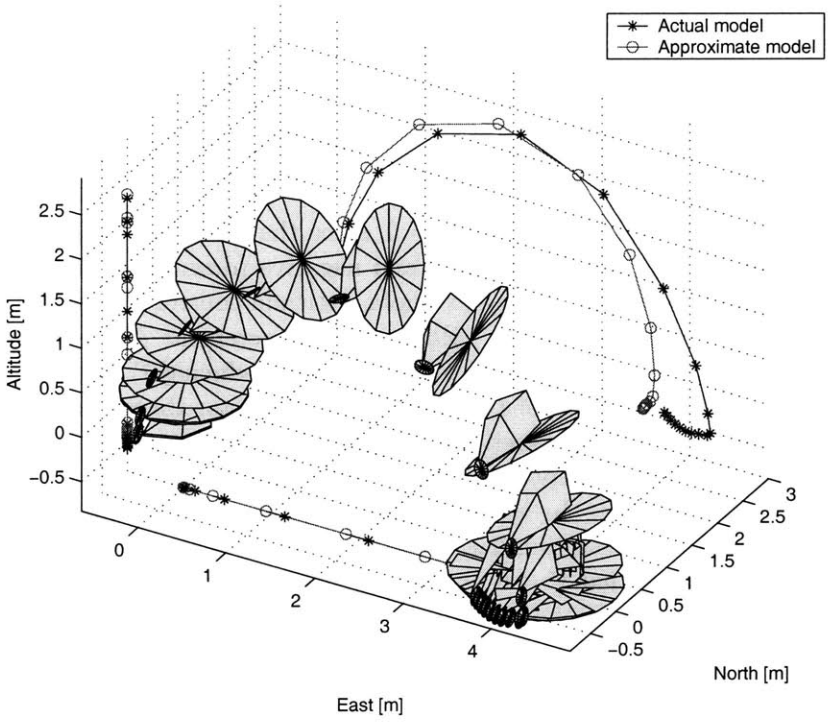

Figure 5-5: Maneuver tracking example.

Chapter 6

Motion Planning in a Dynamic Environment

In this chapter we will present a general-purpose motion planning algorithm for high dimensional nonlinear systems, evolving in complex, dynamic environments. While in the previous chapters we dealt with the generation of trajectories for nonlinear systems, satisfying a number of "internal" constraints, such as those deriving from the system's dynamics, flight envelope and control saturation constraints, in this chapter we will also address the presence of "external" constraints, such as avoidance of obstacles and threats, and other constraints which are not directly related to the vehicle and its dynamics.

The chapter is organized as follows: First, the problem we want to address is discussed, along with the assumptions on the environment and the available information. Second, we introduce the main building block which will be used for exploring and evaluating trajectories: an optimal control law is assumed to be available, for an obstacle-free environment. Third, the proposed randomized algorithm for motion planning in a dynamic environment is presented and discussed, and its performance is analyzed. Finally, the performance of the randomized algorithm is demonstrated on a ground robot example, and on problems of specific interest to the aerospace community, including small helicopter maneuvering in a cluttered environment, and spacecraft attitude slewing.

6.1 Problem Formulation

In this chapter we address Problems 1.4 and 1.5. The main difference of these problems with respect to the steering problems can be found in the presence of the obstacle avoidance constraints (1.3) and the integral constraints (1.4) along with the flight envelope constraints (1.1). While the latter share the symmetry properties of the system's dynamics (i.e. $F(x,u) = F(\Psi_h(x),u)$), the former do not. As a consequence, a dynamic programming formulation such as the one presented in Section (3.3.2), which places the bulk of the computations on an off-line processing phase, is not suitable for this case. While in principle the problem can still be posed in a dynamic programming formulation, an estimate of the optimal cost function has to be recomputed every time the target point or the environment change. In Chapter 3, the optimal cost function was only dependent on the *relative displacement* on the symmetry group; one of the consequences of the presence of obstacle avoidance constraints is the fact that now the optimal cost function will be a function both of the initial conditions *and* of the final conditions (and not only of the relative displacement).

In addition, the simple fact that the obstacle avoidance constraints (1.3) are time varying (i.e. the obstacles are moving), translates into hard real-time requirements for the motion planning algorithm, *even under a perfect information assumption* (stated below). This is a consequence of the fact that the feasibility of the computed solution with respect to the obstacle avoidance constraints has to be checked on (state,time) couples: the solution must be a time-parameterized trajectory, where the time "tags" are of fundamental importance. Because of the finite computation times in physical systems, the motion planning has to be carried out starting from initial conditions at some time in the future (i.e. we need a lead time). If the motion planning algorithm is too late in delivering the solution to the vehicle, that is, if the computation time exceeds the allowed lead time, the late solution is in general not guaranteed to be feasible.

In the following, we will assume that all the obstacle avoidance constraints are available *a priori*, that is, the present position and the future motion of all the obstacles are perfectly known*:

Assumption 6.1 (Perfect knowledge of the environment) *All the obstacle avoidance constraints are available* a priori.

This is admittedly a limiting assumption, since in most practical applications it is not possible to know in advance the motion of the obstacles, especially if the "obstacles" are actually other vehicles, moving according to their own control laws and goals. However, for the time being we will concentrate on this problem, which is already extremely challenging.

Before continuing with the discussion of the motion planning algorithm, we need to introduce some notation. We will say that a couple (x,t) is *collision-free* if it satisfies the obstacle avoidance constraint (1.3). A curve $[t_i,t_f] \ni t \mapsto x(t)$ is said *feasible* if it is an integral curve of the system dynamics given the initial condition $x_i = x(t_i)$, that is if it is possible to find a control input signal u such that $x(t) = \phi_u(t,x_i)$, and if the flight envelope, obstacle avoidance, and integral constraints are satisfied for all times $t \in [t_i,t_f]$. Given an initial condition (x_i,t_i), we say that (x_f,t_f) is *reachable* from (x_i,t_i) is it is possible to find a feasible trajectory connecting (x_i,t_i) to (x_f,t_f). We define the *reachable set* $\mathcal{R}(x_i,t_i)$ as the set of all couples (x,t) which are reachable from (x_i,t_i). Accordingly, given a set $S \subset \mathcal{X} \times \mathbb{R}$, we define:

$$\mathcal{R}(S) := \bigcup_{(x,t)\in S} \mathcal{R}(x,t).$$

The motion planning problem now becomes the problem of finding a feasible trajectory from the initial condition (x_i,t_i) to the desired target x_f. A motion planning algorithm is complete if it returns a feasible solution if and only if there exists some t_f such that $(x_f,t_f) \in \mathcal{R}(x_i,t_i)$, and returns failure otherwise. As we did in the preceding chapters, we will limit our discussion to problems in which the final conditions correspond to a relative equilibrium for the system; as such, the final conditions are effectively parameterized only by the final position on the symmetry group H. Moreover, we will limit the feasible trajectories of the system to a compact subset of the state-space $\mathcal{X}$, and hence a compact subset of the symmetry group H. This can be done easily through the introduction of a "confinement" constraint of the form (1.3). The motivation for such a requirement derives from the necessity of generating uniform distributions of target equilibria on the symmetry group.

*Alternatively, we can also consider conservative bounds on the current and future position of the obstacles, at the cost of increased conservativeness of the computed solution.

Finally, we will assume that the state space $\mathcal{X}$ can be decomposed into the product $\mathcal{Y} \times \mathcal{Z}$, with $\mathcal{Y} \equiv H$. Given a set $S \in \mathcal{X}$, we define $\mu(S)$ as the volume of the canonical projection of the set S on $\mathcal{Y}$.

6.2 Obstacle-free guidance system

This section introduces the main element used to formulate our motion planning algorithm. Our algorithm presupposes the existence of a closed-loop architecture that enables the guidance of the vehicle from any initial conditions to any target location at rest (or at any other desired relative equilibrium). Thus, rather than working with an "open-loop" system, as presented in earlier publications [71, 65, 45], our basic dynamical system is a closed-loop one.

While, admittedly, finding such a guidance law is *per se* a very difficult problem, it also has been the object of an enormous amount of work over the past century: In many cases efficient, obstacle-free guidance laws may be computed analytically [6, 21]. This is the case of system with linear dynamics with a quadratic cost. It also includes numerous cases of aerospace interest such as double or triple integrators with control amplitude and rate limits. In addition, many of these problems, although they may not admit closed-form solutions, may be solved numerically via the approximate or exact solution to an appropriate optimal control problems (see Chapter 1 for a brief discussion and references). Advances in computer power, combined with appropriate plant simplifications (such as the introduction of the Maneuver Automaton model outlined in Chapter 3) make it possible in many cases of practical interest to compute and store an approximate expression for the optimal cost function $J^*(x, x_f)$[†], using for example iterative methods [10].

If the optimal cost function $J^*(x, x_f)$ is known for all initial and final conditions, then it is relatively easy to recover the optimal control policy as a (feedback) policy $\pi(x, x_f)$ that returns at each time instant the control input that minimizes the total cost-to-go to the target [11]. The feedback policy π can be thought of as a function of the state x, parameterized by the destination equilibrium point x_f.

The solution to an optimal control problem in the free space thus provides us with a control policy π that ensures that the system is driven towards a target relative equilibrium (or a neighborhood thereof) in finite time. Moreover, the optimal cost function provides a meaningful measure of the "distance" between the initial conditions and the target.

Along with the optimal control law in an obstacle-fee environment, we assume the availability of a *simulator* of the system, that is a function that is able to predict the evolution of the closed-loop system, and generate the corresponding (space,time) trajectory. Such a simulator can be easily developed from the knowledge of the system's dynamics and of the optimal control law.

6.3 Motion planning in the presence of obstacles

The motion planning algorithm in the presence of obstacles is based on the determination of a sequence of random "attraction points" x_r, and the corresponding control laws $\pi(\cdot, x_r)$,

[†]Because of the symmetry of the obstacle-free problem, the optimal cost will actually be a function only of the *relative* displacement between x and x_f; in Chapter 3 we assumed, without loss of generality, that x_f was at the identity on the symmetry group. Now we will explicitly retain the dependence on the final condition.

that effectively steers the system to the desired configuration while avoiding obstacles. In this way, the obstacle-free solution to an optimal control problem forms the basis for the problem of motion planning in the presence of obstacles. Such an approach casts the location of the equilibrium configuration as a function of time as a "pseudo-control" input for the system. Since the actual control inputs can be computed from the knowledge of the optimal control policy $\pi(\cdot, x_r)$, this means that the low-level control layer (the layer actually interacting with the vehicle) and the high-level, guidance layer are effectively decoupled, while at the same time ensuring full consistency between the two levels. In other words, the output of the guidance layer are control policies, not reference states or inputs. As a consequence, unlike earlier randomized motion planning approaches, the motion planning algorithm can be run at a rate that is much slower than the rate required for the low-level control layer.

Note also that the ideas outlined above in a probabilistic roadmap setting can be seen as a motion planning technique through scheduling of Lyapunov functions, where the Lyapunov functions are defined to be optimal value functions $J^*(\cdot, x_r)$. While the concept is not entirely new in control theory [73, 26, 92], to the author's knowledge, this is the first application to motion planning in a workspace with moving obstacles. A fundamental difference can also be seen in the fact that in our algorithm the ordering of Lyapunov functions is performed on-line, whereas in the references the ordering was determined *a priori*. In other words, in [73, 26, 92], there exists a partial ordering of the Lyapunov function, that is established a-priori: the system is driven through the attraction domains of a pre-established sequence of Lyapunov functions that draws the state closer and closer to the origin. In our algorithm, such a sequence is computed on-line.

Imposing that the motion plan be constructed as a sequence of local optimal control policies, steering the system towards a sequence of attraction points, limits the class of possible motion plans which are output by the algorithm. Hence, the motion planning algorithm we are going to present will not be complete, in the sense that it is not capable of generating all possible feasible trajectories for problem 1.4. On the other hand, we can analyze the completeness of the motion planning algorithm with respect to the class of trajectories it is able to generate.

Given the control policy π, we say that a point (x_f, t_f) is π-reachable from (x_i, t_i) if the closed-loop evolution of the system, under the control policy $\pi(\cdot, x_f)$, with initial conditions (x_i, t_i) is such that a feasible, collision-free trajectory is generated, through (x_f, t_f) (or a neighborhood thereof). The π-reachable set $\mathcal{R}_\pi(x_i, t_i)$ is thus defined as the set of all (state,time) couples which are π-reachable from (x_i, t_i); this set is bounded from below (in the sense of time) by a manifold with the same dimension as the symmetry group H, embedded in the larger space $\mathcal{X} \times \mathbb{R}$ (see Fig. 6.3). Accordingly, we can define the π-reachable set of a set S as:

$$\mathcal{R}_\pi(S) := \bigcup_{(x,t)\in S} \mathcal{R}_\pi(x, t).$$

In the following, we will consider a weaker notion of completeness, limited to trajectories which can be generated by the application of a sequence control policies, from the initial conditions.

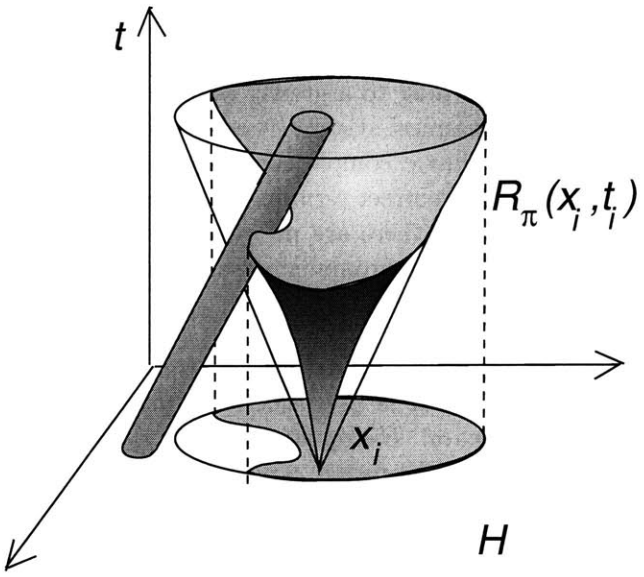

Figure 6-1: Visualization of $\mathcal{R}_\pi(x_i, t_i)$ and its projection on the symmetry group H. The cone represents $\mathcal{R}(x_i, t_i)$.

6.3.1 Overview of the algorithm

The basic idea of the algorithm is the following: starting from the initial conditions (x_i, t_i), we incrementally build a *tree*[‡] of feasible trajectories, trying to explore efficiently the reachable set $\mathcal{R}(x_i, t_i)$.

At each step, we will add a new branch (edge) and a new milestone (node) to the tree. For this purpose, we have to address two points:

- Which node do we want to expand?
- In which "direction" shall we explore?

The first incremental randomized motion planning algorithm was recently introduced by LaValle and Kuffner [72], based on previous work by Kavraki, Svestka *et al.* [57], with the name of Rapidly Exploring Random Tree (RRT). In the original RRT, the answers to the above questions were provided (roughly) in the following way:

- Pick a configuration x_r at random, and choose the node to expand as the *closest* (in the sense of a Euclidean distance) node currently in the tree.
- Then apply a constant input for a time δt in such a way that the final point is moved as close as possible to x_r.

If the resulting trajectory is feasible, then it is added to the tree. The above procedure is iterated until one of the nodes of the tree is "close" enough to the target point x_f.

[‡]A tree is a directed graph, with no cycles, in which all nodes have several outgoing edges, to each of which corresponds another node, called a *child*. A child node can in turn have several other children. All nodes (excluding the root) have one and only one incoming edge, which comes from the *parent*. The root has no parent. A common example of a tree is the directory structure in a computer file system.

While the RRT algorithm proved to be extremely efficient in many difficult problems, as reported in [71, 72, 65], it could not be appropriate in the general case of a dynamical system. Selecting the control inputs according to a greedy policy on the Euclidean distance could result into instability for the dynamical system (it essentially amounts to pure proportional feedback). Also, only a probabilistic completeness proof is available. Even though it is guaranteed that the probability of correct termination of the algorithm converges to one as the number of iterations increases, there are no indications on how fast this convergence will occur. Moreover, in a dynamic environment case, this approach may fail altogether, as the following simple example shows.

Example 6.1 (RRTs and moving obstacles) *Consider a simple one-dimensional, kinematic system (i.e. the control input is the velocity of the system), and assume that the magnitude of the velocity is bounded. The dynamics of the system are described by the ODE* $\dot{x} = u$ *with* $|u| \leq 1$. *Assume that the initial conditions are at* $x = 2/3$, *the desired destination is at* $x = 0$, *and that the control policy* π *derives from the solution of a minimum time optimal control problem. The workspace is bounded between 0 and 1. Finally, assume that at* $x = 1/3$ *there is an obstacle, which is removed at* $t = 1/2$. *The reachable space is depicted in figure 6-2. Applying the RRT algorithm in this case would result in a tree with milestones lying only inside the marked square, thus resulting in an incomplete planner.*

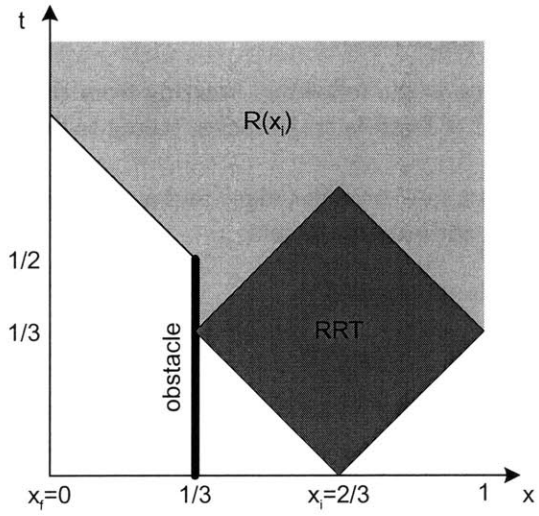

Figure 6-2: One-dimensional dynamic environment example.

In Hsu, Kindel, *et al.* [45] a different approach was introduced, based on the following main steps:

- Choose the node to be expanded at random.
- Apply a random control input for an interval δt.

The algorithm has been proven to be probabilistically complete, even in the presence of moving obstacles. Moreover, its proponents were able to prove performance bounds on the algorithm, provided that at each iteration the reachable set of the set of current milestones (nodes in the tree) is explored uniformly. This is clearly not accomplished in the general

case by the two steps outlined above: A random selection of the inputs from a uniform distribution does not necessarily result into a uniform distribution on the outputs (the location of the new candidate milestone). Moreover, a uniform random choice of the node to be expanded does not result in a uniform exploration of the reachable set of the tree, since the reachable sets from each of the nodes overlap and can have different volumes. As a consequence, the exploration mechanism considered in the proof was not the one outlined in the two steps above, but was instead assumed to be the output of an idealized procedure, denoted as IDEAL-SAMPLE. The random selection of nodes and control inputs was used as an approximation of IDEAL-SAMPLE in the practical implementation. Moreover, the complete randomness of both the main steps in the tree construction results in the generations of trees which appear to lack in "sense of purpose", and do not appear to explore very efficiently the free configuration space (efficient exploration was, on the other hand, the main advantage of the RRT algorithm).

To address these problems, we advocate the fact that the optimal cost function in the obstacle-free environment provides the most appropriate information to address both the issues of node selection and trajectory generation. Using the optimal control function and the corresponding optimal control policy, we will indeed be able to implement an IDEAL-SAMPLE procedure, and achieve the performance bounds in [45]. The key ideas are the following: the correct measure of "distance" is the optimal cost-to-go, and the optimal input can be derived from the associated optimal control policy. Moreover, the optimal (feedback) control policy can be seen as the inversion mechanism which translates the uniform distribution in the output space (i.e. the location of the newly generated candidate milestone x_r) into the corresponding distribution in the input.

In our algorithm we proceed as follows:

- pick a configuration x_r at random, and try to expand all the nodes in the tree in sequence, in order of increasing cost $J^*(x_i, x_r)$ (i.e. starting from the *closest* node, using the measure of distance provided by $J^*(\cdot, x_r)$).

- Apply the optimal control policy $\pi(\cdot, x_r)$, until the system gets to x_r (or a neighborhood of x_r).

As it can be easily recognized, the above steps correspond to the steps in the RRT algorithm, with two fundamental differences: at each iteration, we try to connect the new candidate milestone in turn to each one of the nodes currently in the tree, before discarding it as unreachable from the current tree (in the original RRT only the closest node was tested for reachability). The RRT criterion of testing the closest node translates into the heuristics of testing the nodes in ascending "distance" order. The second main difference is that the optimal cost function in the obstacle-free case is used as a measure of "distance", both in the selection of nodes to expand, and in the computation of the optimal control.

In the next subsections we will discuss more in detail the components of the motion planning algorithm, and how they fit together.

6.3.2 Data structure

First of all, we will discuss briefly the element that characterize each node (also referred to as milestone) and each edge (or branch) in the tree.

Name	Domain
State	$\mathcal{X}$
Time	$\mathbb{R}_+$
CumulativeCost	$\mathbb{R}_+$
LBCost	$\mathbb{R}_+$
UBCost	$\mathbb{R}_+$
IntConstraint	$\mathbb{R}^{n_i}$
NChildren	$\mathbb{N}$

Table 6.1: Information stored at the tree nodes.

Data stored at nodes

The main piece of information stored for each milestone consists of the predicted (state,time) couple that will be attained if the chosen motion plan includes the node at hand; this can be computed by integrating over time the equations describing the dynamics of the system.

Moreover, at each node in the tree we can store data used to compute estimates (lower and upper bounds) on the total cost of motion plans including the node. The total cost of a trajectory (assuming the cost functional is additive, of the form (1.2)) can be split into an accumulated cost and a cost-to-go. The accumulated cost will in general be available, through book-keeping: it is a consequence of decisions made in the past. The cost-to-go is, on the other hand, much more difficult to compute.

However, we can easily compute a lower bound on the cost-to-go: this is given by the value of the optimal cost, in the absence of obstacle. The cost in the presence of obstacles can not be lower than the obstacle-free cost, since the presence of obstacles translates into additional constraints on the problem. If, on the other hand, the optimal trajectory,from the current node to the target, computed in the obstacle-free case, results in a collision-free trajectory even in the presence of obstacles, it is clearly still an optimal trajectory.

The upper bound on the cost-to-go is a priori unknown; as a matter of fact, we don't even know before starting the computations if the problem is feasible or not. As a consequence, we have to initialize the upper bound on the cost-to-go to the symbolic value of infinity. However, if the optimal trajectory from the node to the target is collision free, then the corresponding cost clearly gives an upper bound on the cost-to-go. For a node that can be connected to the target by an optimal trajectory, the lower bound and upper bound coincide. Methods for updating the upper bound on the cost will be discussed in Section 6.3.7.

If we have integral constraints of the form (1.4), we store at nodes the corresponding value of the integral. As a final piece of information, for each node we store the total number of other nodes which are part of its offspring. See Table (6.1) for a summary of the data stored at nodes.

Data stored at edges

While nodes in the tree represent "states" of the system along trajectories, edges can be thought of as representing "decisions" taken in the construction of the motion plan.

As a consequence, the main pieces of information to be stored at edges are the parameters identifying the control law implemented in the transition, namely the next (randomly generated) equilibrium point x_r.

Name	Domain
Target	H
IncrementalCost	$\mathbb{R}_+$

Table 6.2: Information stored at the tree edges.

As a convenience for book-keeping purposes, we can store the incremental cost incurred along the edge. In other words, this corresponds to the difference in the accumulated costs at the edge destination node and at the edge source node.

A summary of the information stored at edges is reported in Table (6.2).

6.3.3 Initialization

The first step in the algorithm is the initialization of the tree. The first node, which will become the root of the tree, contains the initial conditions, projected at some point in the future, which depends on the amount of time which will be allocated to the computation of the motion plan. As an example, if the vehicle at the initial time t_0 is at position x_0, moving at constant velocity v_0, and we devote θ seconds to the computation of a motion plan, the root node must be set at $(x_0 + v_0\theta, t_0 + \theta)$.

The accumulated cost at the root node can be set to zero (all decision variables will have effect later in time, so there is no point with starting with a non-zero value). The lower bound on the cost-to-go can be computed as the value of the optimal cost function from the root state. In order to set the upper bound, we an attempt to reach the target state x_f using the optimal control law $\pi(\cdot, x_f)$. If the resulting trajectory is collision free, the optimal motion planning problem is evidently solved, and the algorithm can return with the optimal solution. Otherwise, the upper bound is set to the symbol ∞ (i.e. it is not yet known whether or not the motion planning problem admits a feasible solution or not), and the tree-building iteration is entered.

6.3.4 Tree expansion step

The main function to be performed at each step in the iteration is the expansion of the tree. The expansion is aimed at adding new nodes, or milestones, to the tree, in such a way that the volume of the reachable space of the tree is increased rapidly.

At each iteration, a new random target configuration x_r is generated, from a uniform distribution. Notice that this requires that the motion planning algorithm concentrates on a compact subset of the configuration space. This limitation can be the result of an obstacle avoidance constraint (such as in the cases in which the vehicle moves within an enclosed environment), or can be artificially imposed by limiting the distance the vehicle can stray from the initial and final points. We also recall that target configurations are assumed to correspond to trim trajectories for the vehicle; in particular, we will concentrate on equilibrium points (i.e. hovering for rotorcraft). This is not a requirement on the planner, but rather depends on the availability of an appropriate optimal control law in the obstacle-free case.

Given the new random candidate milestone x_r , we have to check whether or not it is in the π-reachable set of the current tree. In order to do this, we apply the control policy $\pi(\cdot, x_r)$ starting from the (state,time) initial condition of each node in the tree, until a feasible, collision-free trajectory which reaches x_r (or a neighborhood) at time t_r is

generated. If no such a trajectory is found, then clearly x_r is *not* in the π-reachable set of the current tree, and is discarded. Otherwise (and pending the safety check discussed in the next section) the new trajectory segment is added to the tree, as well as the new milestone (x_r, t_r).

It is easy to see that the function outlined above does indeed result in uniform sampling of the π-reachable set of the current tree (i.e. it is an implementation of an `IDEAL-SAMPLE` procedure on the symmetry group H). The new candidate milestone x_r is generated from a uniform distribution on H, and a thorough test is carried out to check whether or not it is in the π-reachable set of the current tree. The feedback control law $\pi(\cdot, x_r)$ is used as the inversion mechanism that provides the appropriate inputs at each instant in time so that eventually the system will approach uniformly distributed milestones.

The order in which the milestones currently in the tree are checked is as yet left unspecified. A random ordering is acceptable. However, some performance improvements are obtained in simulations (see Section 6.5) if π-reachability of the random candidate milestone x_r is tested from tree nodes in a more efficient way. Typically, in randomized motion planning algorithms, most of the time is spent in checking trajectories for collisions with obstacles, and methods are sought to reduce the number of collision checks [15].

Exploration heuristics

Before a feasible trajectory is found, the emphasis of the algorithm is on *exploration*, that is on the addition on new milestones to the tree which enlarge its reachable set. To enhance exploration, while keeping the number of collision checks to a minimum, it can be convenient to sort the nodes in ascending order of "distance" from x_r, where as a measure of distance we use the value of the optimal cost function in the obstacle-free case $J^*(\cdot, x_r)$. At the cost of the additional computation time required by the sorting of the nodes (and by the evaluation of "distances"), reachability is tested first from the nodes that are "closer" to the candidate milestone, and hopefully more likely to provide a collision-free trajectory. This will result in a lower number of calls to the collision-checking routine (e.g. a routine that evaluates $G(x,t)$ along newly generated trajectories). This exploration heuristics is very closely related to the RRT algorithm , with the only difference that potentially *every node* in the tree is tested during the expansion step, whereas in the RRT algorithm only the *closest node* is tested.

Optimization heuristics

Once a feasible trajectory has been found, the focus of the search shifts from exploration to the *optimization* of the computed trajectory. To enhance the quality of the new additions to the tree, we can sort the nodes in ascending order of total cost to reach x_r. This total cost consists of the accumulated cost up to the generic node (x_i, t_i), plus the cost to go from x_i to x_r, i.e. $J^*(x_i, x_r)$. At the cost of the computation time required to sort the nodes, reachability is tested first from the nodes which will provide the "best" trajectory to the new milestone. This will hopefully result in a better quality of the computed solution. This optimization heuristic is most appropriate when a feasible solution is already available.

As a last detail, every time a new node is added to the tree, the children counter of all the nodes on the path from the new node to the root of the tree must be increased by one.

Algorithm 1: Pseudo-code for the function EXPAND-TREE($Tree$).

1: Generate a random configuration x_r
2: sort the nodes in the tree according to the desired heuristics [random | exploration | optimization].
3: **for all** nodes in the tree (in the order established at the previous step) **do**
4: Generate a trajectory to x_r, using the control policy $\pi(\cdot, x_r)$.
5: Check the generated trajectory for satisfaction of the obstacle avoidance constraints
6: **if** the trajectory is collision-free **then**
7: **return** generated trajectory
8: **end if**
9: **end for**
10: **return** failure

6.3.5 Safety check

Due to the fact that obstacles are moving, checking the absence of collision point-wise in time could not be enough. If we assume bounded accelerations, the reachable set of any collision-free point can be made arbitrarily small (e.g. a point a distance d in front of an obstacle moving with velocity v is collision free, but will not be such d/v seconds in the future). In order to ensure that the tree being built does not include such dead-ends (i.e. milestones with a very small reachable set), before adding a new milestone to the tree we check for its safety over a time buffer τ. We will call τ-safety the property of a milestone of being collision-free over a time τ. Accordingly, we will say that a point (x_f, t_f) is (π, τ) reachable from (x_i, t_i), and belongs to the set $\mathcal{R}^{\tau}_{\pi}(x_i, t_i)$ if it is π-reachable, and τ-safe.

The τ-safety check can be carried out on the predicted evolution of the system at the relative equilibrium corresponding to the new milestone. If possible, and practical, the time buffer τ can be extended to infinity. This is the case, for example, for static environments. Otherwise, τ should be chosen "long enough" that the successful computation of another motion plan is very likely. Assuming that the initial conditions are τ-safe, the τ-safety check ensures that the same safety properties are maintained along the computed motion plan. In the case in which $\tau = \infty$ the τ-safety check translates into hard safety guarantees for the whole motion plan. In cases in which safety cannot be ensured over an infinite time horizon, τ-safety only ensures that the algorithm will always have at least τ seconds to compute a new solution.

6.3.6 Improving performance

As it can be easily recognized, the tree expansion step outlined in Section 6.3.4 above generates trajectories consisting of jumps from equilibrium point to equilibrium point[§], and as such is unlikely to provide satisfactory performance, in terms of the cost (1.2).

However, and building upon previous ideas [73, 26, 92], performance may be restored by realizing that the available guidance policy may not only steer the vehicle from equilibrium state to equilibrium state, but from *any* state to an equilibrium state. This suggests introducing the following step: consider the tree at some point in time and a newly added milestone to the tree (which we will denote as a *primary* milestone). A *secondary* milestone

[§]Or in the general case, from a relative equilibrium to the same relative equilibrium, in a different location on the symmetry group

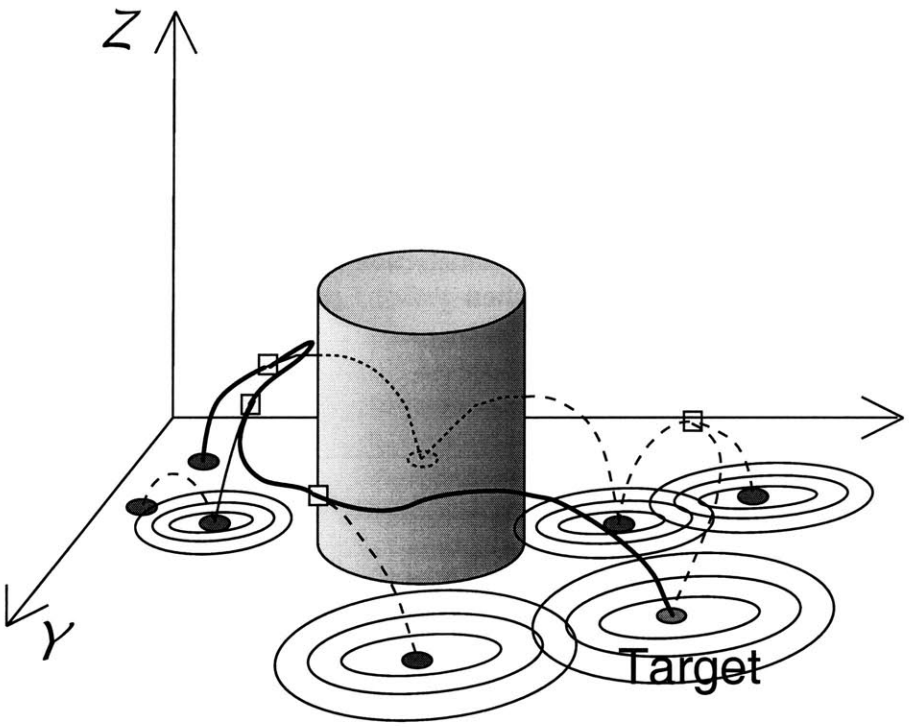

Figure 6-3: Example of generated roadmap (projected on $\mathcal{X}$).

is defined to be any state of the system along the path leading from the parent node in the tree to the newly added milestone. We can split any newly generated edge into $n > 1$ segments at random; the breakpoints will be the secondary milestones, and the endpoint is the primary milestone. Because the vehicle is in motion along the path, the secondary milestones are likely to be at points in the state space that are "far" from the equilibrium manifold.

Note that all secondary milestones, by construction, have a τ-safe primary milestone in a child subtree (see Fig. 17), and hence are themselves τ-safe.

6.3.7 Update on the cost-to-go estimates

Each time a new milestone is added to the tree, a check is made to see if the final target point is in its π-reachable set. In other words, we try applying the control policy $\pi(\cdot, x_f)$ to each newly generated milestone. In the case in which this results in a feasible, collision-free trajectory, we have found sequence of milestones, and control policies, which steers the system from the initial conditions to the final target point along a feasible, collision-free trajectory, and hence the feasibility problem 1.4 is solved.

However, we might be interested in looking for the best possible trajectory satisfying the feasibility problem, and eventually try to approximate the solution of the optimal motion planning problem 1.5.

With this purpose in mind, every time a new feasible solution is found, the estimates of the upper-bound on the cost-to-go of the affected node in the tree are updated. The affected nodes are those in the path from the newly added node to the root. As a consequence, in order to update the upper bound on the costs we climb the tree backwards towards the root:

Algorithm 2: Pseudo-code for the function UPDATE-COST-ESTIMATES($Tree, node, target$).

```
1: node.LowerBound ← J*(node.x, target.x).
2: Generate the obstacle-free optimal trajectory to x_f, using the control policy
   π(·, target.x).
3: Check the generated trajectory for satisfaction of the obstacle avoidance constraints
4: if the trajectory is collision-free then
5:    node.UpperBound ← node.LowerBound.
6:    while node ≠ Tree.root and node.Parent.UpperBound > node.UpperBound +
      node.incomingEdge.cost do
7:       node.Parent.UpperBound ← node.UpperBound + node.incomingEdge.cost
8:       node ← node.Parent
9:    end while
10:   return success
11: else
12:   node.UpperBound ← +∞.
13:   return failure
14: end if
```

at each step we compare the old upper bound at the parent node with the upper bound on the cost following the newly found trajectory. If the latter is smaller that the old upper bound, we update the bound and reiterate from the parent. Otherwise the procedure stops (there is another subtree of the parent which presents a lower upper bound on the cost).

6.3.8 Tree pruning

The upper and lower bounds on the cost to go stored for each tree milestone may be profitably used for pruning the tree and speeding up computations. Recall that the lower bound coincides with the optimal cost-to-go in the obstacle-free case, and the upper bound is equal to the cost of the best trajectory from the milestone to the destination x_f if this trajectory has been found, or $+\infty$ otherwise.

Every time a new feasible solution is found, the upper bounds on the cost-to-go may be updated by climbing the tree backwards along that feasible solution towards the tree root. While performing this operation, it is also possible to look at all the children of the node being updated. If the lower bound on the total cost-to-go for such a children (plus the cost of the corresponding edge) is higher than the upper bound on the cost-to-go for the current node, the corresponding subtree can be safely removed, as it cannot possibly provide a better solution than the one which has just been found.

The end result of such a process is the removal from the trajectory tree of all the provably bad candidates for the "optimal" solution. The trajectory tree, following this pruning process, contain a smaller number of nodes, thus improving the overall computational efficiency. However, it must be kept in mind that tree pruning can only be carried out once a feasible solution has been found, and is of no help before that happens.

6.3.9 Real-time considerations

A significant issue arising from the usage of randomized algorithms for path planning is the distinct possibility of driving the system towards a dead-end due to finite computation

times. The notion of τ-safety was introduced in Section 6.3.5 to prevent such situations to develop.

The τ-safety of the generated plan derives from the fact that all the primary milestones are by construction τ-safe, and all secondary milestones have at least one primary milestones in their sub-tree. Maintaining safety guarantees in the face of finite computation times is particularly important, since the algorithm itself has no deterministic guarantees of success. In the sense outlined above the algorithm will always produce safe motion plans, even in the case in which a feasible trajectory to the target set has not been found.

The time available for computation is bounded by either θ, or by the duration of the current trajectory segment. When the time is up, a new tree must be selected from the children of the current root. If there are none, since every primary milestone is τ-safe, the system has at least τ seconds of guaranteed safety, available for computing a new tree (secondary milestones always have at least one child). If the current root has children, then two cases arise:

- At least one of the children leads to the destination through an already computed feasible solution. If there are more than one such feasible solution, the solution with the least upper bound on the cost-to-go is chosen.

- No feasible solutions have been computed yet. In this case there is no clear indication of the best child to explore. Maintaining the same approach at the basis of the algorithm, the child to descend can be selected randomly, according either to a uniform distribution, or to a distribution weighted on the total number of primary milestones in the sub-children of each tree. In the latter case the selected tree is likely to cover a bigger portion of the reachable set.

6.3.10 Complete algorithm

The flow of the algorithm then proceeds as follows. After the initialization of the tree, a loop is entered, until the target has been reached. As the very first step in each iteration of the loop, an attempt is made to join the current root (the "initial conditions"), to the target configuration, using the optimal control law computed for the obstacle-free case. If the attempt is successful, an optimal solution has been found, and the algorithm terminates successfully.

Otherwise, an inner loop is entered, which tries to expand the trajectory tree. At each iteration of the inner loop, a random configuration is sampled, and an attempt is made, through the `EXPAND-TREE` function, to join a node in the current tree to the new candidate milestone. If this is possible, and results in a τ-safe trajectory (i.e. the candidate milestone is in the (π, τ) reachable set of the current tree), then the generated trajectory is split up into a number of segments, which are added to the tree as new edges, and the corresponding endpoints, which are added to the tree as new (primary or secondary) milestones.

From the newly generated milestones, an attempt is made to reach the final target. If successful, this attempt allows the update of the upper bound estimates on the cost-to-go on the tree nodes. In this case, the tree can be pruned to eliminate nodes in the tree which are guarantee to be bad candidates for further exploration (since they cannot improve on the quality of the solution already computed).

This iteration is carried out until the time allowed for computation has elapsed. This time is given by the time duration of the edge of the tree that is currently being executed,

Algorithm 3: Pseudo-code for the motion planning algorithm.

```
1: Initialize Tree with the initial conditions at time t_0 + θ.
2: loop
3:   if UPDATE-COST-ESTIMATES(Tree, root, target) = success then
4:     Terminate with success
5:   end if
6:   repeat
7:     newTrajectory =EXPAND-TREE(Tree)
8:     if newTrajectory ≠ failure and newTrajectory is τ-safe then
9:       Split newTrajectory and generate primary and secondary milestones
10:      for all new milestones do
11:        UPDATE-COST-ESTIMATES(Tree, node, target)
12:      end for
13:      Prune tree
14:    end if
15:  until Time is up
16:  if Tree.root.UpperBound < ∞ {Feasible solution found} then
17:    find the child of root which gives the least upper bound on the cost-to-go
18:    Tree.root ← best child
19:  else if Root has children then
20:    Choose a child according to a random distribution, weighted with the number of
       children in each subtree
21:  else
22:    propagate root for a time θ in the future {If θ ≪ τ this can be done a large number
       of times}
23:  end if
24: end loop
```

and is therefore not a priori fixed. Note that the iteration can be preempted with no adverse consequence on the algorithm.

At this point, if at least one feasible solution had been found, the subtree of the root corresponding to the least upper bound on the cost is chosen. If no feasible solution has been found yet, a subtree to descend is chosen randomly from a distribution weighted with the total number of milestones in each subtree. If the current root has no children, a new root is created, which corresponds to the predicted root state at some time in the future (this time interval being the computation time allotted for the next main loop iteration).

The rest of the tree is deleted and removed from memory, since it contains only trajectories which can no longer be reached (since it is not possible to go back in time). The main loop iteration is then repeated.

6.4 Analysis

This section aims at analyzing the behavior of the algorithm, proving probabilistic completeness and obtaining performance bounds. This requires additional definitions and assumptions about the environment characteristics. The remainder of the section presents the concepts supporting most of the available results about algorithms based on probabilistic

roadmaps. New definitions are needed to adapt previous results to the motion algorithm presented in this chapter, in particular to account for the closed-loop dynamics, and to better characterize the assumptions about the dynamic environment.

Assumptions on the environment

First of all, we require that for all τ-safe equilibrium points the (π, τ)-reachable set is not "too small", taking into account also computational delays. This property corresponds to the ϵ-goodness of a static workspace [56]. The planning environment (as defined by both the workspace and the control policy) is said to be (ϵ, τ)-good if, for all sets $S_{\tau-\theta} \subset \mathcal{F}$ of $(\tau - \theta)$-safe equilibrium points, the following holds:

$$\mu(\mathcal{R}_\pi^\tau(S_{\tau-\theta})) \geq \epsilon.$$

The proof of algorithm performance relies upon the following properties, which will be assumed to be true for the system under consideration.

Let β be a constant in $(0, 1]$; define the *β-lookout* of $S \subset \mathcal{F}$ as

$$\beta\text{-lookout}(S) := \{p \in S | \mu(\mathcal{R}_\pi^\tau(p) \setminus S) \geq \beta\mu(\mathcal{R}(S) \setminus S)\} \tag{6.1}$$

We require that the dimensions of the β-lookout of any set should not be "too small"; this property is called (α, β)-expansiveness [46]: Given two constants α, β in $(0, 1]$, the environment is said (α, β)-expansive if for all sets $S \in \mathcal{F}$ the following holds:

$$\mu(\beta\text{-lookout}(S)) \geq \alpha\mu(S). \tag{6.2}$$

Algorithm performance

Consider the initial condition (x_0, t_0), and assume it is an equilibrium point (if not, generate a primary milestone using the algorithm presented in the previous section). Define the *end-game region* $E \subset H$ as a region such that all equilibrium points contained in it can be connected without collisions to the desired destination x_f using the policy $\pi(\cdot, x_f)$, for all times t. Then, if the environment is (α, β)-expansive, and the desired destination x_f is contained in the reachable set $\mathcal{R}(x_0, t_0)$, it can be shown that the probability of the algorithm returning a feasible trajectory connecting x_0 to x_f approaches unity exponentially fast. The results that will be presented in the following are based on Hsu *et al.* [45], with a few minor modifications to address the usage of the policy π in the exploration process, as opposed to random, piecewise constant inputs.

First of all, we need to characterize the expected number of look-out points in a tree. We have the following:

Lemma 6.1 (Number of look-out points [45]) *In a tree with r milestones, the probability of having k look-out points is at least $1 - k\exp\left(-\alpha\lfloor\frac{r}{k}\rfloor\right)$.*

Proof: First of all, cut the number r of milestones to an integer multiple of k, and call it $\tilde{r}$. Naturally, we have that the probability of having k lookout point in the original tree is greater than it is on the reduced tree; moreover $\tilde{r} \leq k\lfloor r/k \rfloor$. Now we can split the $\tilde{r}$ milestones in k sequences of equal length. The probability of having k lookouts points is greater than the probability that each and every one of the sequences contains a lookout point.

Since the EXPAND-TREE function samples uniformly in the space, each milestone in the sequence is a look-out point with respect to the preceding set milestones with probability α, and the probability of the i-th sequence to contain no lookout points is at most $(1-\alpha)^{\tilde{r}/k}$. Hence the probability that at least one of the k sequences does not contain any lookout points is at most $k(1-\alpha)^{\tilde{r}/k}$. From the negation of this event, we get the probability of having a lookout point in each sequence, that is at least $1-k(1-\alpha)^{\tilde{r}/k} \geq 1-k\exp(-\alpha\tilde{r}/k)$. ■

The following step is to quantify the size of the (π,τ)-reachable set given the presence of at least k lookout points in the tree:

Lemma 6.2 (Size of reachable set) *If a sequence of milestones contains k lookout points, the volume of its (π,τ)-reachable set is at least $1-\exp(-\beta k)$.*

Proof: Let us look at the volume of the complement of the reachable set. If the sequence of milestones does not contain any lookout point, then the size of the unreachable set is at most 1. Each time a new lookout point is found, the size of the unreachable set is reduced at least by a factor $(1-\beta)$, by definition of lookout point. Hence after k lookout points the volume of the unreachable set is at most $(1-\beta)^k \leq \exp(-\beta k)$, and the result follows. ■

Theorem 6.1 (Performance of the randomized algorithm [45]) *A sequence of r (primary) milestones contains a milestone in the end-game region E with probability at least $1-\gamma$, if:*

$$r \geq \frac{k}{\alpha}\ln\frac{2k}{\gamma} + \frac{2}{\mu(E)}\ln\frac{2}{\gamma} \tag{6.3}$$

where $k := (1/\beta)\ln(2/\mu(E))$.

Proof: Split the sequence of milestones into two sub-sequences of r_1 and r_2 milestones, respectively. If the number of milestones in the first sequence is greater than $k = 1/\beta\log(2/\mu(E))$ then the (π,τ)-reachable set of the first sequence has an intersection of volume at least $\mu(E)/2$ with the end-game region E. Let us call event A the event that the number of lookout points is greater than k. This event occurs with a probability at least:

$$\Pr(A) \geq 1 - e^{-\alpha\lfloor\frac{r_1}{k}\rfloor}. \tag{6.4}$$

If we want to make this probability at least $1-\gamma/2$ we need:

$$r_1 \geq \frac{k}{\alpha}\log\frac{2}{\gamma}.$$

Assume that the intersection of the (π,τ)-reachable set of the first sequence of milestones with the end-game region is $\mathrm{Vol}\,E/2$. Then the probability that at least one of the milestones in the second sequence will be in the end-game region (event B) is at least:

$$\Pr(B) \geq 1 - e^{-r_2\mu E/2}. \tag{6.5}$$

In order to make this probability at least $1-\gamma/2$, we need:

$$r_2 \geq \frac{2}{\mu(E)}\log\frac{2}{\gamma}.$$

If events A and B occur, then one of the milestones in the complete sequence is in the endgame region. We have that, if $r \geq r_1 + r_2$:

$$\Pr(A \wedge B) = \Pr(A)\Pr(B) \geq (1 - \gamma/2)^2 \geq 1 - \gamma, \tag{6.6}$$

which proves the result. ∎

To our knowledge, the algorithm presented in this chapter is the first one to which Theorem 6.1 fully applies, with the definitions and under the assumptions given earlier in this section, since the (π, τ)-reachable set is indeed uniformly sampled. In some sense the most significant contribution of this chapter is to propose to shift the search for reachable milestones from an *open-loop* process, whereby exploration is done by randomly sampling the controls available to the system to a *closed-loop* process, whereby the exploration is done by randomly (and uniformly) sampling the milestones, and the obstacle-free guidance system then chooses the controls leading the vehicle from its current state to that milestone. In this sense the guidance law can in fact be interpreted as a practical implementation of the ideal inversion mechanism proposed in [45].

Moreover, we are able to recover, by sorting nodes according to "distance", the levels of performance shown in practice by RRT algorithms. As a consequence, we are able to achieve the probabilistic completeness and the formal performance bounds in a dynamic environment proven in [45] for an idealized procedure, and at the same time we can exploit the rapid exploration capability of RRTs: in this sense the algorithm presented in Section 6.3 recovers all the best properties of its predecessors.

The performance bounds that can be obtained for this algorithm establish only its theoretical soundness, but cannot be used for obtaining an explicit estimate of the probability of successful termination, since α, β, and ϵ cannot be computed easily for non-trivial environments. However, the result in Theorem 6.1 establishes the theoretical soundness of the algorithm. An interesting fact is that the computational complexity (in terms of convergence rate of the probability of correct termination) does not depend on the dimension of the state space, or on the number of obstacle directly. Instead, it depends on parameters that in some sense quantify the "geometric complexity" of the environment.

Note also that using secondary milestones does not adversely impact the main results on probabilistic completeness (when evaluated on the number of *primary* milestones) , since they can only increase the (π, τ)-reachable set of the whole tree. In addition, secondary milestones have been found to help the convergence of the algorithm when close to the endgame region, and enhance the overall quality of the resulting trajectory (i.e. the trajectory is "faster", or in general less expensive with respect to the assigned cost).

6.5 Application Examples

In the next sections we present three examples which show the power and the flexibility of the proposed algorithm. We consider first a linear system, then a system endowed with a control architecture patterned after the hybrid control structure described in Section 3.1.4, and finally a dynamic system which evolves on the rotation group. All algorithms have been implemented in (soft) real time on a C++ on a Pentium II 300 MHz machine running Linux, using the LEDA library [141, 93]. Comparisons will be presented on the computation times required by the following variations of incremental randomized planners:

Algorithm A : Only one node, chosen randomly from the current tree, is tested for expansion at each EXPAND-TREE step. This corresponds roughly to the algorithm in [45].

Algorithm B : Only one node, chosen as the "closest" one to the candidate milestone, is tested for expansion at each EXPAND-TREE step. This corresponds to the RRT algorithm [72], with the difference that the obstacle-free optimal cost-to-go is used to compute "distances".

Algorithm C : All nodes are tested, in random order.

Algorithm D : All nodes are tested, in increasing order of "distance". This corresponds to the full-fledged implementation of the algorithm in Section 6.3.

The inputs are always chosen according to the optimal control policy in the obstacle-free case. Random control inputs are not appropriate for most of the dynamical systems we are considering, since they could lead easily to instability (for example, a helicopter is an open-loop unstable system). If distances are measured according to the optimal cost function in the obstacle-free case, a greedy search for the control generation (as in the RRT algorithm) does provide the optimal control. Moreover, in all cases any additional computation time available after the first feasible solution is found is used to optimize the solution.

The statistical data are referred to a data set of one thousand simulation runs for each example if not otherwise specified.

6.5.1 Ground robot

In this section, we are interested in minimum time motion planning for a planar system with equations of motion

$$\begin{aligned} \ddot{x}_1 + \dot{x}_1 &= u_1 \\ \ddot{x}_2 + \dot{x}_2 &= u_2. \end{aligned} \tag{6.7}$$

The magnitude of each control u_1 and u_2 is assumed to be bounded by u_{max}. Although this system model is quite simple, it is a good representation of the ground robots used by Cornell University team to win the RoboCup-2000 contest [30, 52]. The following control law is adapted from the same references.

Minimum-time, minimum-energy control law

For any one axis, let the initial position and velocity be x_0 and v_0; the final (equilibrium) conditions are characterized by a desired final position x_f and zero velocity. The minimum time maneuver from origin to destination for each of the degrees of freedom (assuming a general maximum control intensity $u_{\max}$) is a bang-bang control law [21] given by

$$\begin{aligned} u(t) &= U \quad for \quad 0 < t < t_1 \\ u(t) &= -U \quad for \quad t_1 < t < t_1 + t_2. \end{aligned} \tag{6.8}$$

The sign of the initial control value U can be determined through the switching function:

$$\Delta_0 := \begin{cases} x_0 - x_f + v_0 - u_{max} \log(1 + v_0/u_{max}) \text{ for } v_0 \geq 0 \\ x_0 - x_f + v_0 + u_{max} \log(1 - v_0/u_{max}) \text{ for } v_0 < 0 \end{cases}. \tag{6.9}$$

If the initial conditions are such that $\Delta_0 \geq 0$ then $U = -u_{max}$, and $U = u_{max}$ otherwise.

The time length of the two bang-bang segments can be determined as follows:

$$\begin{aligned} t_1 &= t_2 - C/U \\ t_2 &= \log(1 + \sqrt{1 - \exp(C/U)(1 - v_0/U)}) \end{aligned} \tag{6.10}$$

with $C = x_0 + v_0 - x_f$.

The policy π used to control the vehicle described by (6.7) is then defined as follows: Considering the two degrees of freedom x_1 and x_2, the "slowest" axis is determined first, and the corresponding time optimal control is applied. Let t^*_{min} be the minimum time corresponding to that axis.

The other, fastest axis is then controlled using a minimum "effort" solution, by solving the minimum time problem using the equations (6.8), with $U = \pm\gamma u_{\max}$, and by iterating over the parameter $\gamma \in (0, 1)$ until $t_{min} = t^*_{min}$.

Fixed and moving spheres

The randomized path planning has been tested in several examples, including cases with both fixed and moving obstacles, and in general proved to be very fast and reliable.

The first example involves navigating the ground robot through a set of obstacles represented as spheres in the configuration space, as shown in Fig. 6-5. In the tests, both fixed spheres and spheres moving at a constant random speed were considered, with no significant statistical difference in the data sets. The pictures reported in the following only depict the fixed sphere case (for ease of representation).

This example was very easily handled by the randomized algorithms. A feasible solution was found by all algorithms in less than 20 ms in 50% of the cases, with no significant differences in performance. A feasible solution was found by all algorithms in all test cases. A summary of the average time required to find the first feasible solution t_{feas} and its standard deviation is reported in Table 6.3.

The cost of the trajectory that is eventually executed has been analyzed. In this example, the lower bound on the cost is 11.3 seconds (this is the cost of an optimal trajectory in the obstacle-free case; note that there does not exist a unique optimal trajectory, but many trajectories minimize the cost). In all cases but for Algorithm 2 an optimal trajectory was found in at least 50% of the test cases. On average, the cost of the solution was less than 5% higher than the lower bound, showing that the algorithms do indeed provide a solution that is almost optimal. In this scenario the algorithms A and C, relying completely on randomization (as opposed to RRT-like heuristics) seemed to fare better than the others. A summary of the average cost of the solution t_f provided by the algorithms and its standard deviation is also reported in Table 6.3.

Examples of the traces of trajectory trees computed during simulation runs are reported in Figure 6-5. In the figures the heavy line represents the trajectory that is eventually executed (i.e. the best computed solution), while the lighter lines represent the other trajectories (edges) in the trajectory tree. The little squares represent primary milestones, and the little circles represent secondary milestones. Milestones which can be connected to the target by the obstacle-free optimal control law are filled in a darker color.

In this case the environment is "open" and all the randomized algorithms can achieve extremely high levels of performance, even in the presence of moving obstacles.

Algorithm	Average(t_{feas}) [ms]	St. Dev. (t_{feas}) [ms]	Average(t_f) [s]	St.Dev. (t_f) [s]
A	17.68	10.10	11.52	0.29
B	16.34	7.93	12.03	0.68
C	18.12	10.01	11.55	0.31
D	17.3	8.57	11.86	0.55

Table 6.3: Ground robot moving amidst fixed spheres: simulation results.

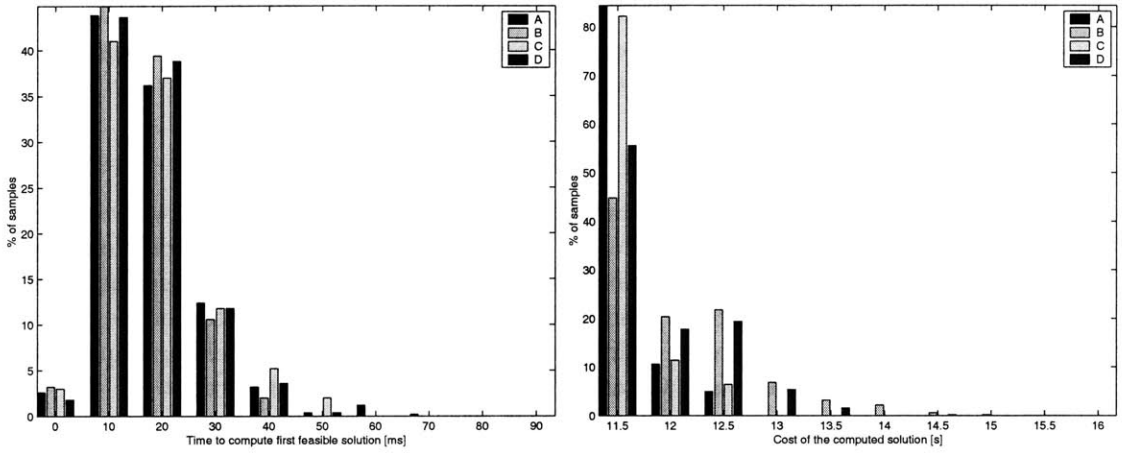

Figure 6-4: Ground robot moving amidst fixed spheres: histogram of the computation times required to find a feasible solution (left) and of the cost of the computed solution (right).

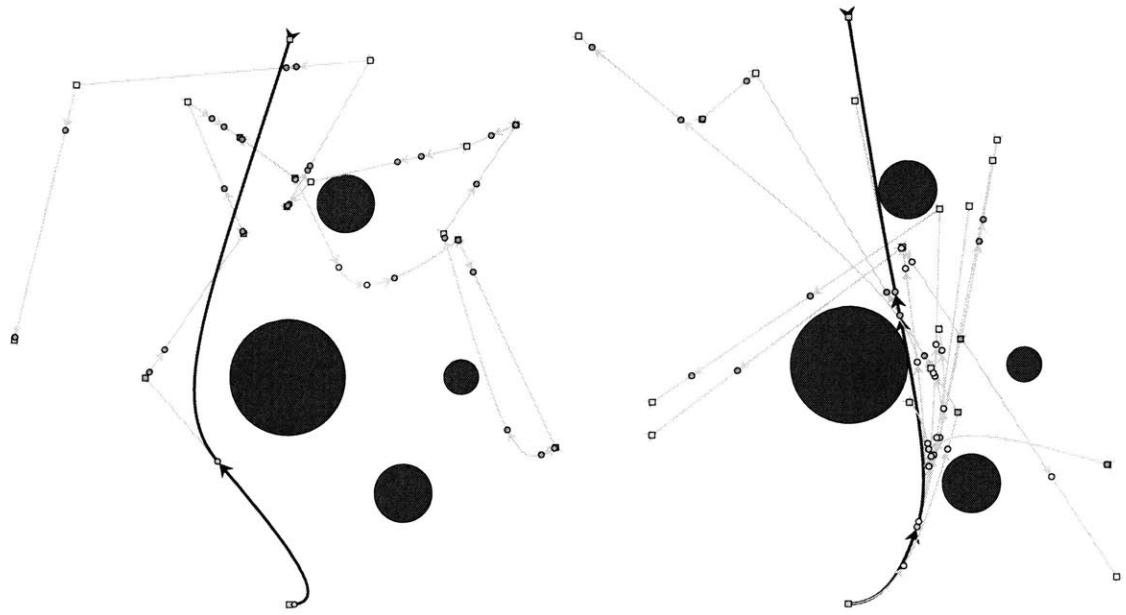

Figure 6-5: Ground robot moving amidst fixed spheres: trajectory tree traces. Algorithm A (left) and Algorithm C (right).

Labyrinth

In the second example the robot must navigate a maze composed of rectangular walls. The robot starts from the bottom left corner and must go to the top right corner in a square domain (see Figure 6-7). This problem is much more difficult than the previous one, since now the environment has a very low expansivity (i.e. the environment presents several relatively "narrow passages"). This results in much longer times required to compute the first feasible solution: for this example, computation times are of the order of one second for most of the algorithms. However, in many cases Algorithm A took a much longer time (i.e. 30 seconds on average) to find a feasible solution, and the robot never reached the target in the allotted time of one minute. This is a consequence of the fact that algorithm A, while complete, can be quite inefficient in the exploration of environment with narrow passages. The best performance was obtained with algorithms C and D.

Sliding Doors

In the third and last example the robot must go through moving openings in two walls. The walls are 40 meters apart, and have "doors" 10 m wide. Both the walls slide along their longitudinal axis according to a harmonic law. The frequencies of the two oscillations are 0.5 rad/s for the wall at the bottom, and 0.25 rad/s for the wall at the top of the picture. The amplitude of the harmonic motion in both the cases is 40 meters, yielding a maximum door velocity of 20 m/s, twice as much as the maximum velocity of the robot.

This scenario was expected to be a difficult challenge for the randomized planners: however, most of the algorithms were able to deal with it quite effectively. Algorithm A was slow in finding a feasible solution (it took an average time of over 40 seconds), but the quality of the resulting plans was consistently good. Algorithm C and D were again the best performers, finding a feasible trajectory within a few seconds. Moreover, the average cost of the plans generated by Algorithm C is within 22% of the lower bound on the optimal cost, that is 11.39 seconds.

Algorithm	Average(t_{feas}) [s]	St. Dev. (t_{feas}) [s]	Average(t_f) [s]	St.Dev. (t_f) [s]
A	30.47	29.39	—	—
B	2.329	9.10	11.18	1.31
C	0.89	0.78	9.44	0.96
D	0.70	0.6	11.03	11.03

Table 6.4: Ground robot moving in a maze: simulation results.

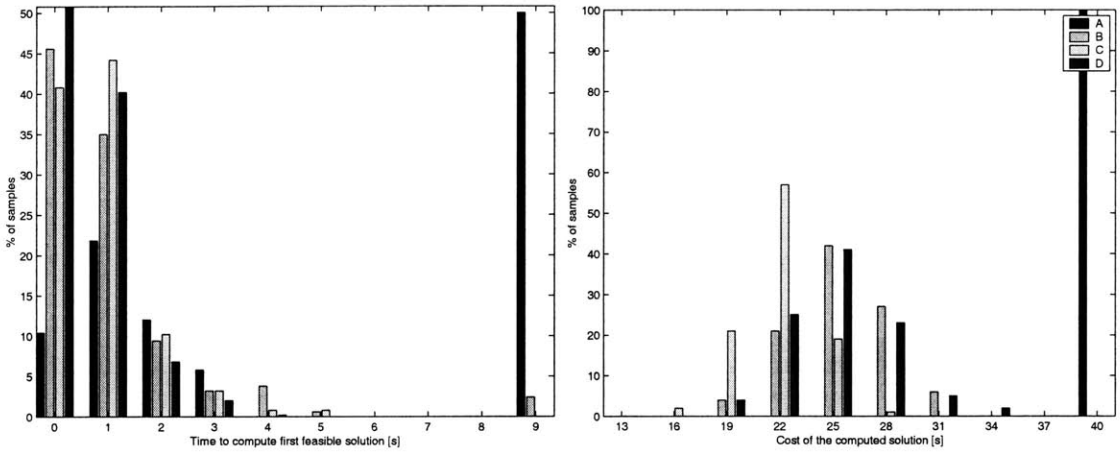

Figure 6-6: Ground robot moving in a labyrinth: histogram of the computation times required to find a feasible solution (left) and of the cost of the computed solution (right).

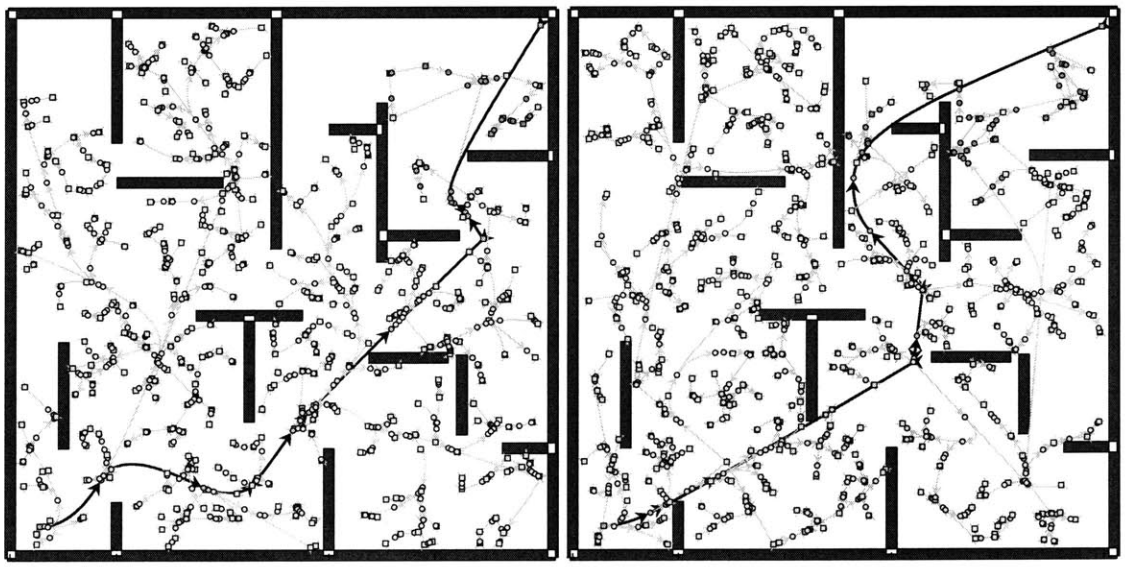

Figure 6-7: Ground robot moving in a labyrinth: trajectory tree traces. Algorithm B (left) and Algorithm D (right).

Algorithm	Average(t_{feas}) [s]	St. Dev. (t_{feas}) [s]	Average(t_f) [s]	St.Dev. (t_f) [s]
A	42.67	27.25	14.95	6.47
B	14.53	25.68	18.78	5.96
C	3.12	13.16	13.87	5.54
D	1.29	8.43	17.35	5.88

Table 6.5: Ground robot moving through sliding doors: simulation results.

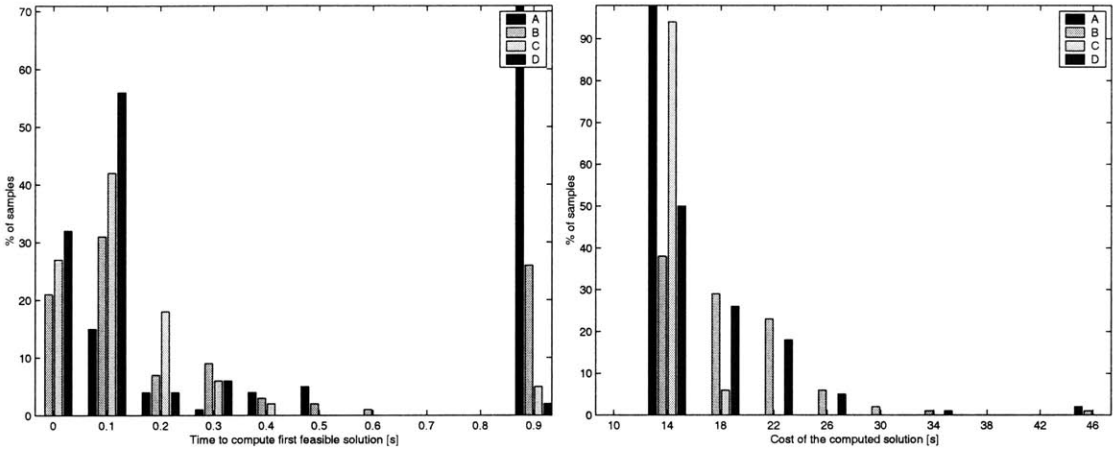

Figure 6-8: Ground robot moving in through sliding doors: histogram of the computation times required to find a feasible solution (left) and of the cost of the computed solution (right). The openings in the wall are shown at the position at the moment in which they are crossed by the robot.

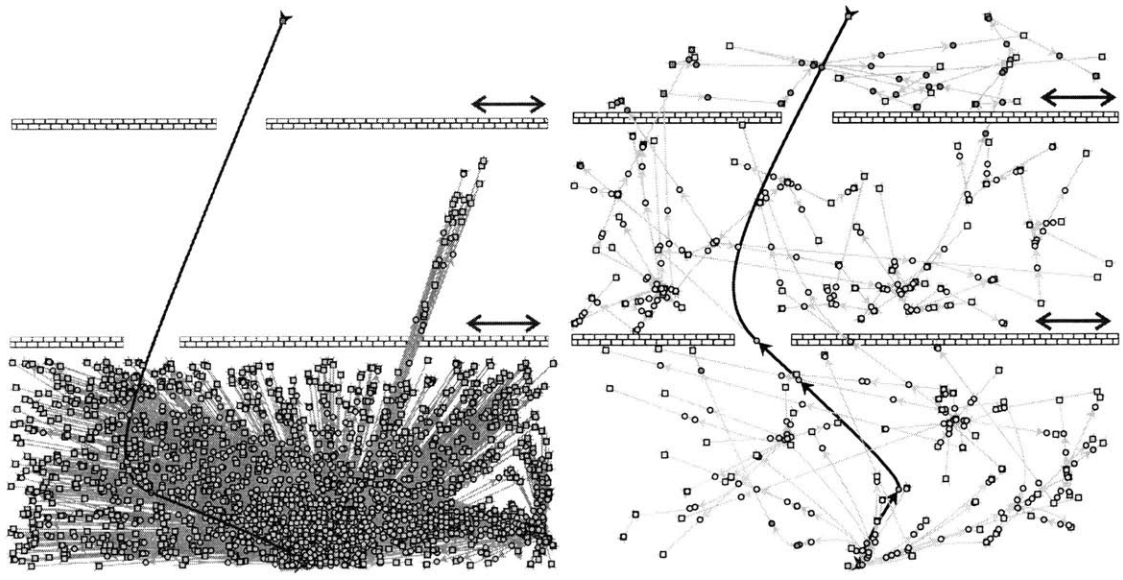

Figure 6-9: Ground robot moving through sliding doors: trajectory tree traces. Algorithm A (left) and Algorithm D (right).

6.5.2 Small autonomous helicopter

One of the prime motivations for the development of the algorithm presented in this chapter is the path planning task for a small autonomous helicopter.

This section presents simulation results for a test case involving a small autonomous helicopter. The simulations rely upon a fully non-linear helicopter simulation, based on the model presented in Section 2.3.3 and in the references therein. The motion planning algorithms operating on the nominal maneuver automaton structure presented in Chapter 3 are complemented by a tracking control law as discussed in Chapters 4 and 5 to ensure tracking of the computed trajectory.

The planner was tested using the same scenarios as for the ground robot examples. The output of the simulations was scaled, in such a way as to provide a meaningful comparison of the two cases. The cost function used in all the examples is the total time needed to go to the destination. The initial conditions are at hover, heading due East (i.e. to the right in the pictures).

Fixed and moving spheres

The first example involves navigating the helicopter through a set of fixed spheres. As in the case of the ground robot, this example was very easily handled by the proposed planners: the average time required for computation of a feasible solution is of the order of a few tenths of a second (see Table 6.6).

Labyrinth

The second example involves navigating the helicopter through a maze of fixed walls. This example proved to be the extremely difficult for all of the algorithms under consideration: a reason for the poor behavior can be seen in the fact that the maneuvering capabilities of the helicopter inside the maze are very limited.

Within a simulation time of two minutes, Algorithm A did not find a feasible trajectory in 71% of the cases, Algorithm B in 58% of the cases, and Algorithms C and D in 52% and 53% of the cases, respectively.

Sliding doors

In the last example the helicopter must go through moving openings in two walls (see Fig. 6.5.2).The walls are 40 meters apart, and have "doors" 10 m wide. Both the walls slide along their longitudinal axis according to a harmonic law. The frequencies of the two oscillations are 0.5 rad/s for the wall at the bottom, and 0.25 rad/s for the wall at the top of the picture. The amplitude of the harmonic motion in both the cases is 40 meters, yielding a maximum door velocity of 20 m/s, twice as much as the maximum velocity of the helicopter.

This scenario proved to be very challenging for the randomized planners. Algorithm A failed to find a feasible solution within two minutes in the totality of the simulations run (one thousand). Algorithm B did better, finding a feasible solution within two minutes in 10% of the cases (in which the feasible trajectory was found extremely quickly, in about 2.5 seconds). Algorithms C and D on the other hand always succeeded in finding a solution, even though it took about 50 seconds on average. The total cost of the solutions found was slightly above one minute on average.

Algorithm	Average(t_{feas}) [s]	St. Dev. (t_{feas}) [s]	Average(t_f) [s]	St.Dev. (t_f) [s]
A	0.365	0.390	16.44	2.30
B	0.117	0.060	18.36	3.59
C	0.181	0.124	17.05	2.58
D	0.183	0.132	17.26	2.42

Table 6.6: Helicopter moving amidst fixed spheres: simulation results.

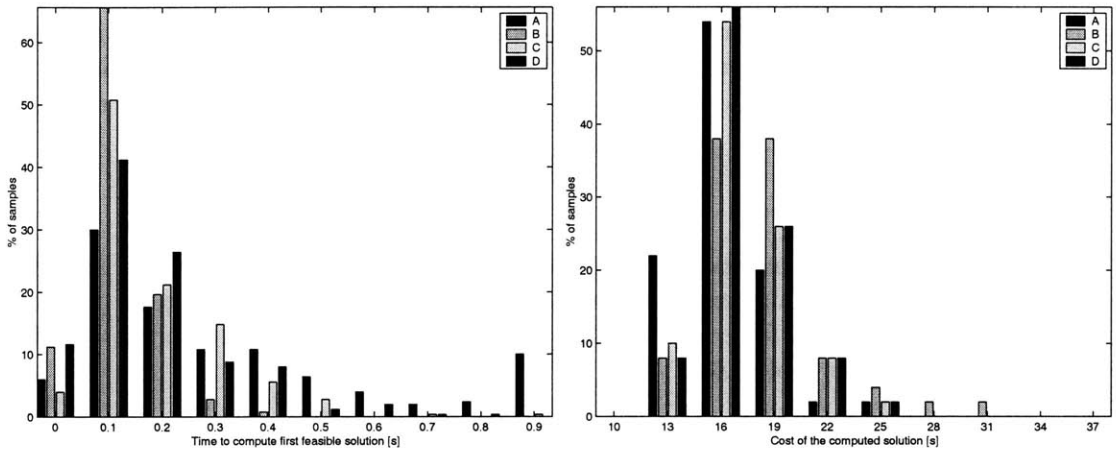

Figure 6-10: Helicopter moving amidst fixed spheres: histogram of the computation times required to find a feasible solution (left) and of the cost of the computed solution (right).

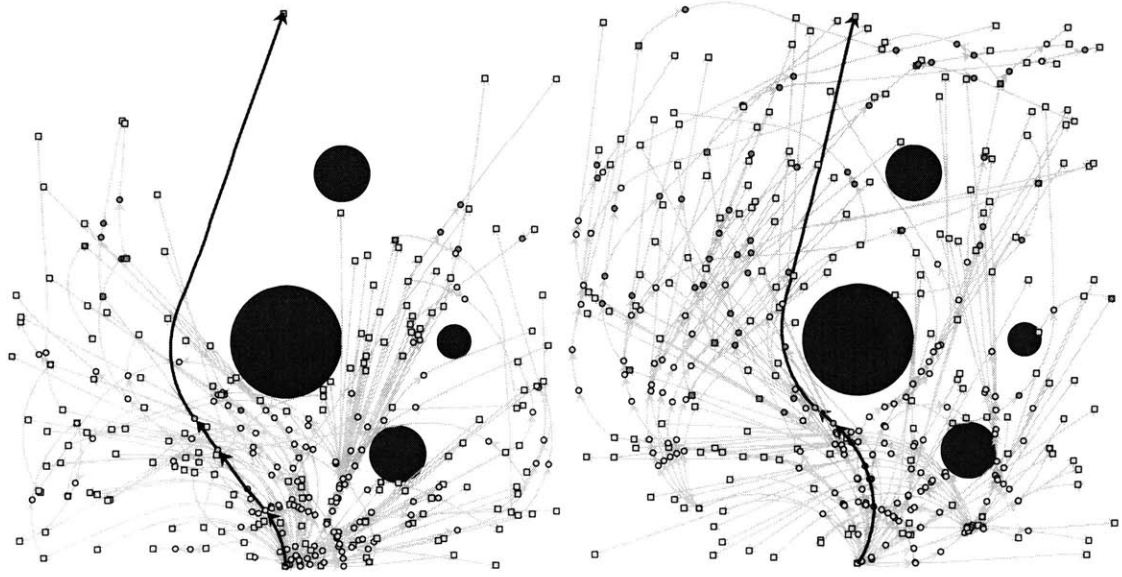

Figure 6-11: Ground robot moving in a labyrinth: trajectory tree traces. Algorithm A (left) and Algorithm D (right).

Algorithm	Average(t_{feas}) [s]	St. Dev. (t_{feas}) [s]	Average(t_f) [s]	St.Dev. (t_f) [s]
A	106.24	33.62	118.70	15.13
B	72.19	56.52	85.40	18.07
C	75.22	48.48	82.87	18.70
D	70.49	53.55	81.30	19.62

Table 6.7: Helicopter flying inside a maze: simulation results. If the computation times exceeded the allotted time of 120 seconds, the value of 120 seconds was taken into account in the statistics.

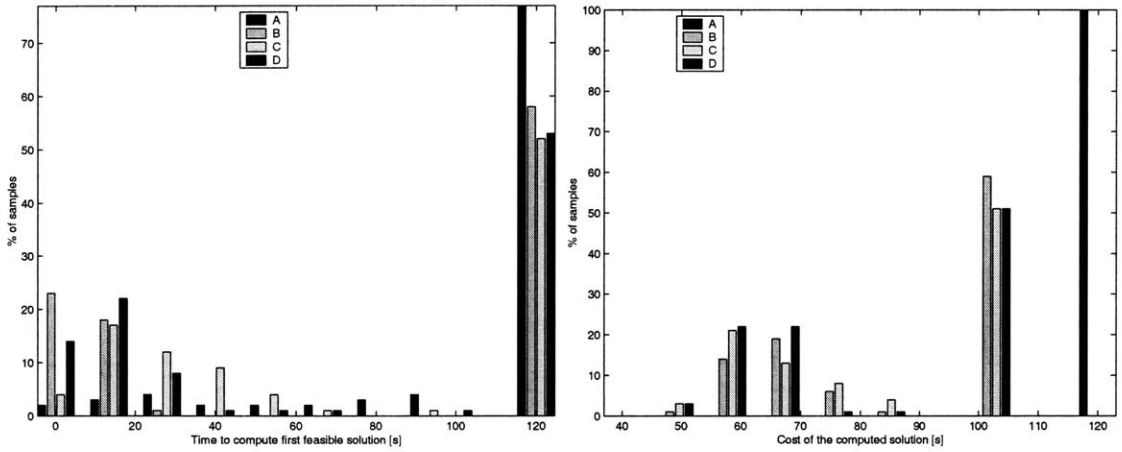

Figure 6-12: Helicopter flying inside a maze: histogram of the computation times required to find a feasible solution (left) and of the cost of the computed solution (right).

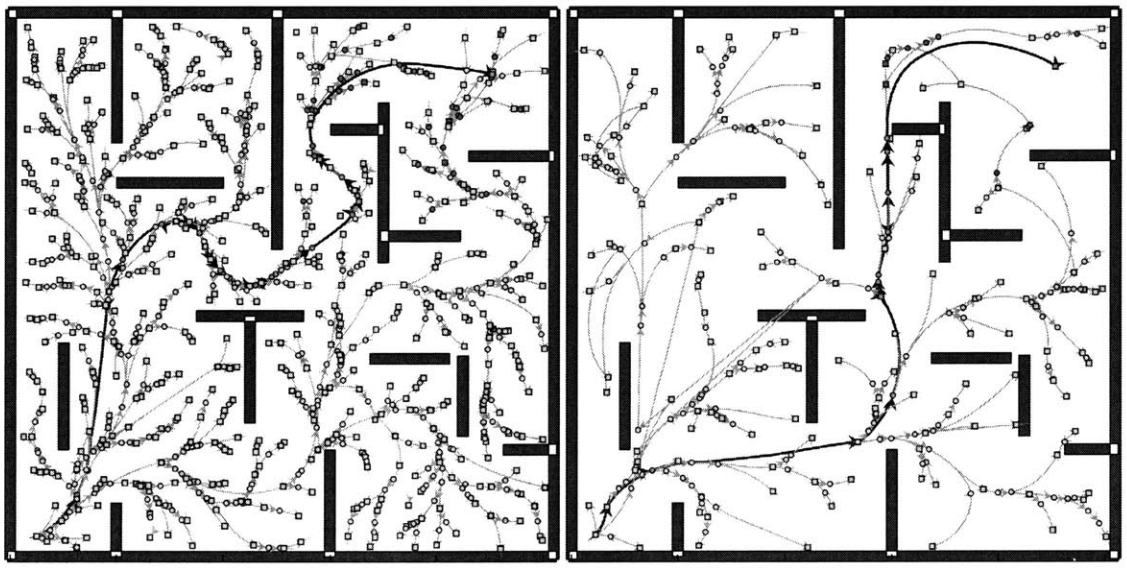

Figure 6-13: Ground robot moving in a labyrinth: trajectory tree traces. Algorithm B (left) and Algorithm D (right).

Algorithm	Average(t_{feas}) [s]	St. Dev. (t_{feas}) [s]	Average(t_f) [s]	St.Dev. (t_f) [s]
A	—	—	—	—
B	108.26	35.58	118.04	28.78
C	48.89	42.33	64.83	25.79
D	52.44	38.51	68.18	32.59

Table 6.8: Helicopter flying through sliding doors: simulation results. If the computation times exceeded the allotted time of 120 seconds, the value of 120 seconds was taken into account in the statistics.

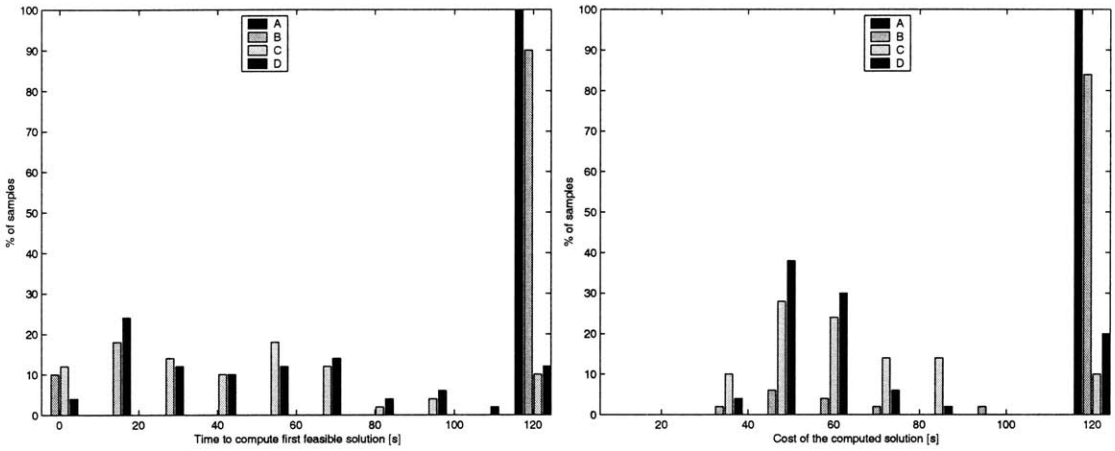

Figure 6-14: Helicopter flying in through sliding doors: histogram of the computation times required to find a feasible solution (left) and of the cost of the computed solution (right). The openings in the wall are shown at the position at the moment in which they are crossed by the helicopter.

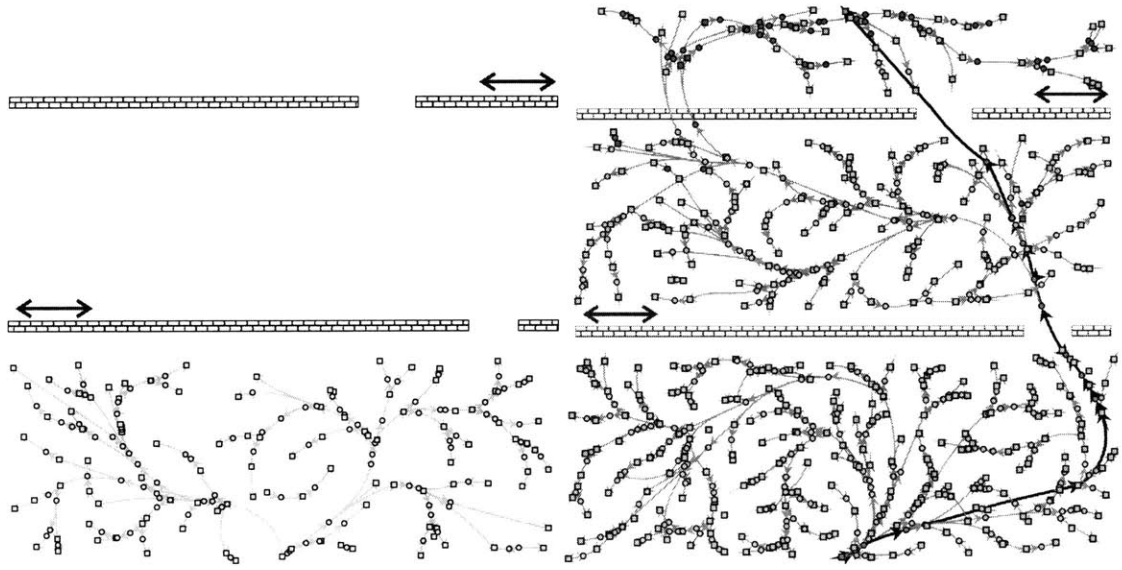

Figure 6-15: Helicopter flying through sliding doors: trajectory tree traces. Algorithm B (left) and Algorithm D (right). The RRT-like algorithm on the left can get "locked" by the first computed trajetories.

Discussion of the results

From the analysis of the simulation results, both in the ground robot example and in the helicopter example, we can conclude that, all the tested algorithms perform equally well in "simple" cases, with very open environments. As the difficulty of the test scenarios increases, it is apparent that algorithms C and D (two versions of the algorithm proposed in this chapter) performed better than algorithms A and B, both in terms of computation time required to solve the feasibility problem, and in terms of the quality of the computed solution.

While algorithm A does have probabilistic completeness guarantees in the case of dynamic environments (as opposed to algorithm B, which does not), it suffers from inefficient exploration, as it can easily be recognised from Figure 6.5.1.

Algorithm B, while performing well in static environments, shows its limits in the case of dynamic environments (in fact, it is not probabilistically complete in this case). This algorithm is characterized by a very efficient exploration of the workspace: however it tends to get locked in the first computed trajectories which, even though "close" to the objective, could not result in feasible trajectories to the target. It is believed that this is the reason of the behavior noted in the case of the helicopter flying through sliding doors.

Algorithms C and D provided the best results among the algorithms tested. Even though the performance characteristics are very similar, algorithm D seems to find the first feasible solution in a shorter time than C. The reason for this can be found in the fact that by sorting the nodes in order of increasing distance from the candidate milestone, the efficient exploration characteristics of RRTs are recovered, while ensuring probabilistic completeness and exponential convergence to one of the probability of correct execution, even in a dynamic environment. On the other hand, algorithm C generally resulted in solutions with a lower cost: this can be seen as another effect of the fact that RRT-like algorithms tend to get locked in the first solutions they compute, and do not explore other options.

6.5.3 Spacecraft Attitude Motion Planning

As a final example, an additional possible application of the motion planning algorithm presented in this chapter is briefly introduced. A key enabling technology that could lead to greater spacecraft autonomy is the capability to autonomously and optimally slew the spacecraft between attitudes while operating under a number of celestial and dynamic constraints. For example, when planning a spacecraft slew maneuver, great care has to be taken in order to protect sensitive science or navigation sensors (e.g. telescopes, star trackers) from direct exposure to the Sun and other bright bodies. These constraints can be pointwise in time, or they can be cast as integral constraints (e.g. constraints deriving from thermal control). The task of finding an attitude trajectory that meets all the constraints is a challenging one: in fact, attitude maneuver planning has traditionally been performed on the ground.

The motion planning algorithm outlined in this chapter can be applied directly to the problem of spacecraft slewing: We can identify the configuration space G as the rotation group $SO(3)$; in the case in which we can model the spacecraft as a rigid body, the space $\mathcal{Y}$ is the three-dimensional linear space of angular velocities expressed in body fixed coordinates. The guidance policy π can be chosen from several approaches that have been proposed for designing obstacle-free attitude control policies. In this example, an obstacle-free attitude

control laws derived by Bullo and Murray [25] was used. This guidance law provide a large set of attraction for equilibrium points, and is appropriate for the case in which the spacecraft is equipped with reaction wheels, providing independent torques on three axes.

Assume that the objective of the (obstacle-free) guidance law is to slew the spacecraft to an attitude R_d, with zero angular rate. The guidance law is based on the definition of a Lyapunov function composed of an term penalizing the attitude error $\phi := R_d^T R$, and a term penalizing angular rates:

$$V(R, \omega, R_d) = R_d^T R + \frac{1}{2}\omega^T J \omega.$$

The derivative of the above function can be made negative definite in a set such that $V(R, \omega, R_d) < 2$, by setting the control moments as follows:

$$u = -\operatorname{Skew}(R_d^T R)^\vee - \omega, \tag{6.11}$$

where $\operatorname{Skew}(M) := 1/2(M - M^T)$, and the notation $(\cdot)^\vee$ indicates the inverse of the "hat" operator introduced before.

It is important to remark that the control policy outlined in Eq. (6.11) is not optimal with respect to any meaningful cost: as a consequence, it is not possible to lower-bound the cost-to-go, and tree pruning cannot be performed in this case. Moreover, the control law is not globally stabilizing. When generating milestones, an additional constraint to be satisfied will then be the requirement that the tree node being expanded lie in the in the domain of attraction of the candidate milestone [91].

The formulation of obstacles, and the "collision checking" function used to ensure feasibility of a planned trajectory, can be easily extended to include a wide range of pointwise in time constraint checking. Moreover, the fact that the algorithm involves building a tree rooted at some initial condition, allows easy implementation of integral constraints, by simple bookkeeping at the tree nodes and edges.

In the following example, the case of a deep-space spacecraft, cruising in the space between the orbits of Mars and Jupiter, is considered. The spacecraft carries a telescope with a fixed boresight in body axes. In exactly the opposite direction is the boresight of a star tracker. In order to protect the optics, the telescope boresight cannot be less than 20 degrees off the Sun, whereas the star tracker boresight must stay at least 40 degrees away from any bright object, including the Sun, as well as the Earth, Mars, and Jupiter. An attitude slew is commanded, for which a direct path would result in a violation of the constraints. A feasible solution was found by the proposed algorithm in about 1.1 seconds on average (standard deviation 0.6 seconds). An example of a planned trajectory is depicted in Fig. (6.5.3).

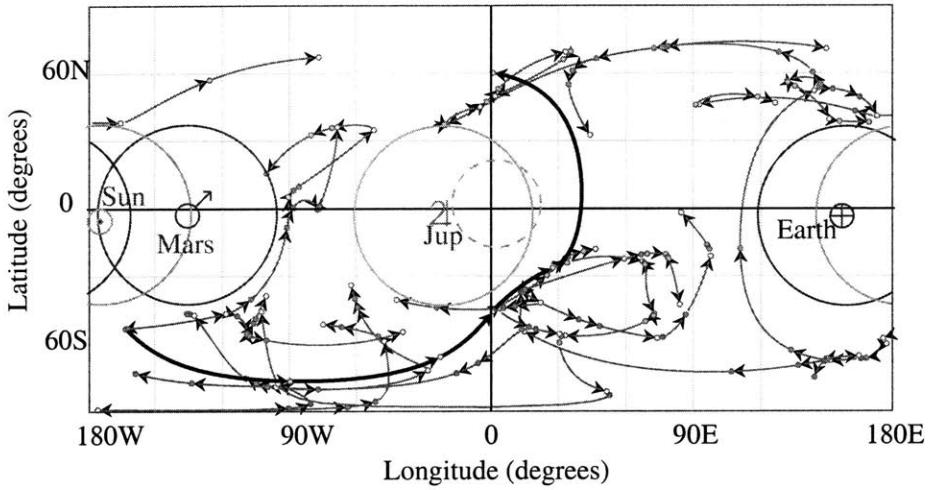

Figure 6-16: Attitude motion planning example: trace of the trajectory tree, for the boresight of the star tracker. The solid circles represent constraints for the star tracker, the dashed circle represents the constraint for the telescope.

Chapter 7

Conclusions

In this thesis we have presented a set of tools which can be used together as a full package to build a complete, consistent and integrated guidance and control system for autonomous vehicle motion planning.

The guidance and control framework presented in the dissertation is based on a computational approach, which relies on a relatively expensive off-line computation phase to provide the capability to compute almost optimal control laws in real-time. Moreover, the proposed framework builds on the fundamental characteristics of the vehicle's dynamics to achieve a high level of computational efficiency, in terms of the trade off between performance loss and computational tractability.

In this chapter we will first summarize the tools presented in the dissertation, and then present some ideas for further research.

7.1 Summary

In Chapter 2 we have introduced a framework for the study of mechanical control systems. In particular, we focused on mechanical control system which evolve on Lie groups. We have then looked closely at the concepts of symmetries and relative equilibria, which are among the fundamental properties of the dynamics of many mechanical control systems, especially vehicles.

In Chapter 3 we have introduced the main core of the framework proposed in this dissertation, the Maneuver Automaton. The Maneuver Automaton is introduced as a new dynamical system, which can model, exactly and in a consistent fashion, an important subset of the dynamics of the vehicle. This is achieved through a quantization of all feasible trajectory, and the selection of a finite number of "trajectory primitives". Two classes of trajectory primitives are identified as trim trajectories (or relative equilibria) and maneuvers. The dynamical system arising from the controlled switching from one of these primitives to another is shown to provide a very attractive alternative to traditional representations of the dynamics, based on differential equations, when computational efficiency is sought. In particular, the solution of optimal control problems is discussed. Moreover, some insight into the trade-off between the complexity of the Automaton and the optimality of the generated trajectories is provided. Examples on several systems, both academic examples and aerospace vehicles are presented and discussed. In particular, in this chapter the process of constructing a Maneuver Automaton model for the approximate dynamics of a small autonomous helicopter is discussed.

In Chapter 4 we provide indications on how to extend the nominal Maneuver Automaton model to provide consistency with the system's dynamics, in the presence of disturbances, uncertainties. The concepts of invariant error, and invariant tracking are introduced. This leads to the definition of invariant sets on trim trajectories and Poincaré maps on maneuvers, and to the derivation of sufficient nesting conditions to ensure the consistency of the Automaton, which hence becomes a Robust Maneuver Automaton. A simple academic example is presented.

In Chapter 5 the precepts of the previous chapter are used to construct an invariant tracking controller for a small autonomous helicopter model. The tracking controller is able to provide asymptotic tracking almost globally, for all feasible trajectories of the helicopter. The proposed control law improves on other available tracking control law for the same system because it avoids all artificial singularities introduced by minimal attitude representations, thus allowing smooth tracking of acrobatic maneuvers. Moreover, the tracking controller provides bounded tracking in the presence of bounded disturbances; based on the estimates invariant sets, it is possible to construct a Robust Maneuver Automaton, taking into account the terms in the helicopter dynamics which were neglected in Chapter 3.

Finally, in Chapter 6 we present a general algorithm for motion planning in a dynamic environment (i.e. an environment characterized by moving obstacles). The algorithm is applicable to all dynamical systems (including Maneuver Automata), for which an optimal control law is available, in an obstacle-free environment. By relaxing the deterministic completeness requirement on the algorithm in favor of probabilistic algorithm, computational tractability is achieved, while maintaining formal performance guarantees (albeit in a probabilistic sense). The main contribution of this chapter are the introduction of the optimal obstacle-free control law as a means to direct exploration and provide a measure of "distance" between configurations. This allows the resolution of technical problems in the randomized motion planning literature, and results in better overall performance of the algorithm. It is shown how the use of an optimal control law in the obstacle-free case is helpful in looking for (sub)optimal solutions. Moreover, real-time implementation issues are addressed. Even in the case of finite computation times, it is shown that under certain conditions the algorithm always provides safe (collision-free) solutions, even though a feasible trajectory leading to the target has not been found yet. Examples on a wide range of systems of interest to the aerospace community are presented and discussed, showing the advantages of the proposed formulation.

7.2 Future directions

In this section we will present some ideas for extension of this work to other problems.

7.2.1 Robust replanning versus robust tracking

In Chapter 4 we have provided the conditions under which a Maneuver Automaton can be made robustly consistent. In other words, we have provided the conditions under which the problem of motion planning on the nominal dynamics, as represented by the Maneuver Automaton, can be decoupled from the problem of robustly tracking the generated reference trajectory.

In this sense the motion planning algorithm applied to the Maneuver Automaton results into a robust motion planning algorithm: the generated trajectory can be tracked to within some known bounds, even in the presence of bounded, worst-case disturbances.

However, in several cases it might be convenient to replan the trajectory based on the current state, and on the disturbance history. Disturbances are not always "worst case", but can instead be helpful. For example a tail wind might allow higher speeds during the cruising phase of an aircraft mission: trying to slow down to track the "nominal trajectory" is likely to be waste of resources.

A possible way to implement a replanning capability in the Maneuver Automaton structure is to introduce a "pseudo-control" η_u, which can smoothly change the nominal position on the symmetry group h. For example, during the execution of a trim trajectory, with initial conditions (q, h), the action of such a pseudo-input would be that of changing the evolution of h from $\dot{h} = h\eta_q$ to $\dot{h} = h(\eta_q + \eta_u)$. The objective of the pseudo-input would be that of "pushing" the nominal trajectory towards trajectories with a smaller cost, while at the same time ensuring that the physical state stays inside the invariant sets of the updated nominal trajectory. This corresponds to "bending" the pipes describing the invariant sets about trajectory primitives, and leads us to the next point.

7.2.2 Maneuver flexibility

A limitation of the Maneuver Automaton framework in its current state is that the maneuvers are very rigidly defined, both in terms of nominal trajectory, and perhaps more importantly in terms of "commitment" required from the control system.

In order to reduce the "rigidity" of the maneuvers in the library, maneuvers can be augmented by a set of parameters describing the actual implementation of the maneuver. Such parameters could, for example, dictate the "level of aggressiveness" of the maneuver. Other parameters could represent the effect of varying environmental conditions (e.g. air density) on the nominal profile of the maneuver.

If we call the parameters λ, the control decision process will then become an optimization over λ as well (and hence we see that the computational complexity of the control computation will grow). However, optimizing over a finite set of parameters describing some aggregate effect on the outcome of the maneuver, still results in a relatively easy optimization problem.

Another limitation of the current structure of the Maneuver Automaton is that, once a maneuver has been initiated, it cannot be aborted. Indeed, the option of aborting a maneuver can be extremely valuable, especially in an unknown or uncertain environment, and in the presence of threats. The difficulty here is that in order to maintain the maneuver automaton consistency, checks must be made to ensure that the system can switch from the trajectory primitive being executed, to some other primitive while staying inside invariant sets. In order to choose the optimal maneuver to be executed next, it is important to identify all possible switch destination, associated with the allowable displacement on the symmetry group.

In order to avoid an excessive number of checks, at the expense of storage space, a set of possible switch destination can be encoded for each sample point in a maneuver nominal trajectory. Another alternative is to provide pre-computed "branching points" along maneuvers (this amount essentially to increasing the number of maneuvers). Clearly, these suggestions increase the capabilities of the vehicle, but at the same time they increase the computational complexity of the motion planning problem to be solved on-line: this is another example of the trade-off between performance and computational tractability that is at the basis of the Maneuver Automaton design. It would be interesting to research alternative approaches, or possibly improvements on the Maneuver Automaton structure,

that allow this kind of flexibility with a relatively small increase in the complexity of the related motion planning algorithms.

7.2.3 Sensor-based motion planning

A very strong, and possibly limiting, assumption on the randomized motion planning algorithm introduced in Chapter 6, is the assumption of perfect information on the environment. We needed this assumption in order to make sure of the future feasibility of all the branches in the trajectory tree.

If this assumption is not granted, every time new information is available, the affected branches in the trajectory tree must be either checked again for feasibility, or discarded altogether.

As a matter of fact, since the algorithm is in practice very fast and efficient, it is possible possible to recompute the whole motion plan every time new information is available.

Early investigation, conducted by the author on the Robocup testbed developed at Cornell University [30], showed that cold restart of the motion planning algorithm is not advisable, since it can result in random wandering of the robot before a complete, feasible solution to the target is computed. This suggests that some information about previous runs of the motion planning algorithm should be carried over to the following runs (warm restart).

In order to reuse the whole trajectory tree, all of its branches must be checked for feasibility again: this can be very expensive from the computational point of view. A better choice seems to be the use as a seed for the next motion planning iteration of only the "best trajectory" in the tree, chosen according to the criteria in Section 6.3.9.

7.2.4 Multiple vehicle operations

The problem of designing autonomous systems becomes even more challenging, and potentially rewarding, when it is desired to coordinate several independent vehicles, communicating over a network with limited bandwidth and/or reliability.

Formation flight

A first problem involving multiple vehicles consists of formation flight. Flying in close formation can be beneficial for fixed-wing aircraft because it allows the reduction of the negative effect of the finite aspect ratio of wings, by reducing the induced drag on the aircraft following the leader of the formation (this is the reason why most migratory birds fly in formation).

Close regulation of the relative position is needed to avoid crashes, and reap the benefits of the induced drag reduction. Moreover the leader must avoid maneuver which cannot be executed by the formation as a whole without the risk of collisions.

An augmented version of the Maneuver Automaton could be beneficial in addressing this problem: an Automaton whose trajectories represent feasible trajectories for the formation as a whole would be clearly beneficial in devising good reference trajectories. Moreover, by providing definite bounds on the position errors of all the aircraft in the formation, the risks of collisions can be mitigated. Last, since the Maneuver Automaton provides a very compact representation of the state of all the vehicles in the formation, it can be very attractive in situations were communication capabilities are limited.

An interesting problem would also be the combination of a Maneuver Automaton for the whole formation and Maneuver Automata for each individual aircraft. Conditions should be given on the compatibility between the two kinds of Automata, and on the requirements in the switching between the two (corresponding to the action of joining/splitting from the formation).

Multiple vehicle coordination

A possibly even more interesting topic would involve the coordination of several vehicles, considered as separate and independent entities (as opposed to the formation, which can be seen as a single entity, with many degrees of freedom). Each of the vehicles would have to gather information, process it, and come up with decisions on actions to perform.

In order to coordinate with other vehicles, it is important to communicate plans and intentions to other vehicles, possibly over a bandwidth limited or unreliable wireless network. The Maneuver Automaton framework can be appropriate, as a means to increase the semantic value of the symbols that are transmitted over the network. With a very small sequence of maneuver indices and coasting times, any vehicle can encode a possibly very complicated plan for future actions, and accordingly receive the intentions of other vehicles. The current state of each vehicle can be communicated by transmitting just the trajectory primitive index and the current position on the symmetry group. Since the Maneuver Automaton evolution is asynchronous, it is required to send updated on each vehicle's state only when necessary, and not at a constant rate.

7.2.5 Optimal selection of trajectory primitives

In Section 3.4.1 we examined the problem of optimal selection of trajectory primitives for a very simple mechanical system. While the simple result does provide some insight, it would be interesting to research more general cases. In particular, it would be interesting to study the optimal quantization of kinematic systems on Lie groups, as a first step toward the analysis of mechanical control systems on Lie groups.

Bibliography

[1] R. Abraham, J.E. Marsden, and T. Ratiu. *Manifolds, Tensor Analysis, and Applications*, volume 75 of *AMS*. Springer Verlag, New York, NY, second edition, 1988.

[2] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[3] R. Alur, C. Courcoubetis, T. Henzinger, and P. H. Ho. Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In R. L. Grossman, A. Nerode, A. P. Ravn, and H. Rischel, editors, *Hybrid Systems*, volume 736 of *Lecture Notes in Computer Science*, pages 209–229. Springer-Verlag, 1993.

[4] V.I. Arnold. *Mathematical Methods of Classical Mechanics*, volume 60 of *GTM*. Springer Verlag, New York, NY, 2nd edition, 1989.

[5] Z. Artstein. Stabilization with relaxed controls. *Nonlinear Analysis, Theory, Methods and Applications*, 7(11):1163–1173, 1983.

[6] M. Athans and P. Falb. *Optimal Control*. McGraw-Hill, 1966.

[7] F. Austin, G. Carbone, M. Falco, H. Hinz, and M.Lewis. Game theory for automated maneuvering during air-to-air combat. *AIAA Journal of Guidance, Control and Dynamics*, 13(6):1143–1149, November-December 1990.

[8] J. Barraquand, B. Langlois, and J.C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems, Man and Cybernetics*, 22(2):224–241, 1992.

[9] R. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.

[10] D. P. Bertsekas and J.T. Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA, 1996.

[11] D.P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, Belmont, MA, 1995.

[12] J. T. Betts. Survey of numerical methods for trajectory optimization. *AIAA Journal of Guidance, Control, and Dynamics*, 21(2):193–207, March-April 1998.

[13] A.M. Bloch, P.S. Krishnaprasad, J.E. Marsden, and R.M. Murray. Nonholonomic mechanical systems with symmetry. *Archives of Rational Mech. An.*, 136:21–99, 1996.

[14] A.M. Bloch, N.E. Leonard, and J.E. Marsden. Stabilization of mechanical systems using controlled lagrangians. In *IEEE Conference on Decision and Control*, 1997.

[15] R. Bohlin and L. Kavraki. A lazy probabilistic path planner for single query path planning. Submitted to the International Journal of Robotics Research. Available at `http://www.cs.rice.edu/CS/Robotics/publications.html` at the time of writing, 2000.

[16] S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory*, volume 15 of *Studies in Applied Mathematics*. SIAM, Philadelphia, PA, 1994.

[17] D. P. Boyle and G.E. Chamitoff. Autonomous maneuver tracking for self-piloted vehicles. *Journal of Guidance, Control and Dynamics*, 22(1):58–67, 1999.

[18] M. S. Branicky. *Studies in Hybrid Systems: Modeling, Analysis, and Control.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, 1995.

[19] R.W. Brockett. On the rectification of vibratory motion. *Sensors and Actuators*, 20:91–96, 1989.

[20] R.W. Brockett. Minimum attention control. In *Proc. IEEE Conf. on Decision and Control*, pages 2628–2632, 1997.

[21] A. E. Bryson and Y. C. Ho. *Applied Optimal Control.* Hemisphere Publishing, New York, 1975.

[22] F. Bullo. *Nonlinear Control of Mechanical Systems: A Riemannian Geometry Approach.* PhD thesis, California Institute of Technology, 1998.

[23] F. Bullo. Kinematic controllability and decoupled trajectory planning for underactuated mechanical systems. Submitted to the IEEE Transactions on Robotics and Automation. Available at `http://motion.csl.uiuc.edu/ bullo/papers` at the time of writing, November 2000.

[24] F. Bullo, N. E. Leonard, and A. D. Lewis. Controllability and motion algorithms for underactuated Lagrangian systems on Lie groups. *IEEE Trans. Aut. Control*, 45(8):1437–1454, August 2000.

[25] F. Bullo and R. M. Murray. Tracking for fully actuated mechanical systems: A geometric framework. *Automatica*, 35(1):17–34, 1999.

[26] R. R. Burridge, A. A. Rizzi, and D. E. Koditscheck. Sequential decomposition of dynamically dexterous robot behaviors. *International Journal of Robotics Research,*, 18(6):534–555, June 1999.

[27] J. Canny. *The Complexity of Robot Motion Planning: ACM Doctoral Dissertation Award.* MIT Press, 1988.

[28] C.R.Hargraves and S.W.Paris. Direct trajectory optimization using nonlinear programming and collocation. *AIAA Journal of Guidance, Control, and Dynamics*, 10(4):338–342, 1987.

[29] M.A. Dahleh and I.J. Diaz-Bobillo. *Control of Uncertain Systems: a Linear Programming Approach.* Prentice-Hall, Edgewood Cliffs, NJ, 1995.

[30] R. D'Andrea, T. Kalmár-Nagy, P. Ganguly, and M. Babish. The Cornell robot soccer team. In P. Stone, editor, *RoboCup-00: Robot Soccer World Cup IV*, Lecture Notes in Computer Science. Springer, 2001.

[31] L.E. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516, 1957.

[32] H. Edelsbrunner. *Algorithms in Combinatorial Geometry.* Springer-Verlag, Berlin, 1987.

[33] N. Elia and S.K. Mitter. Stabilization of linear systems with limited information. *IEEE Trans. Aut. Control*, 2001. To appear.

[34] N. Faiz, S.K. Agrawal, and R.M. Murray. Trajectory planning of differentially flat systems with dynamics and inequalities. *AIAA J. Of Guidance, Control and Dynamics*, 24(2):219–227, March-April 2001.

[35] M. Fliess, J. Lévine, Ph. Martin, and P. Rouchon. Sur les systèmes non linéaires différentiellement plats. Technical Report 317, C.R. Acad. Sci. Paris, 1993.

[36] E. Frazzoli, V. Gavrilets, B. Mettler, M. Piedmonte, and E. Feron. Aggressive maneuvering of small autonomous helicopters: A human-centered approach. *International Journal of Robotics Research*, 2001. To Appear.

[37] R.A. Freeman and P.V. Kokotović. Inverse optimality in robust stabilization. *SIAM Journal on Control and Optimization*, 34(4):1365–1391, July 1996.

[38] V. Gavrilets, B. Mettler, and E. Feron. Nonlinear dynamic model of a small-size acrobatic helicopter. In *Proc. of the AIAA Conf. on Guidance, Navigation and Control*, Montreal, Canada, 2001.

[39] J.M. Godhavn. Nonlinear tracking for underactuated surface vessels. In *IEEE Conference on Decision and Control*, 1996.

[40] J.M. Gonçalves, A. Megretski, and M.A. Dahleh. Global stability of relay feedback systems. *IEEE Trans. Aut. Control*, 46(4):550–562, April 2001.

[41] J. Hauser and R. Hindman. Manoeuvre regulation from trajectory tracking: feedback linearizable systems. In *Nonlinear Control Systems Design 1995. A Postprint Volume from the 3rd IFAC Symposium*, volume 1, pages 269–274, Oxford, UK, 1996. Pergamon.

[42] J. Hauser, S. Sastry, and G. Meyer. Nonlinear control design for slightly nonminimum phase system: Application to V/STOL aircraft. *Automatica*, 28(4):665–679, 1992.

[43] T. Henzinger, P. H. Ho, and H. Wong-Toi. Algorithmic analysis of nonlinear hybrid systems. *IEEE Trans. Aut. Control*, 43(4):540–554, 1998.

[44] D. Hsu, L.E. Kavraki, J.C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *Proceedings of the 1998 Workshop on Algorithmic Foundations of Robotics*, Houston, TX, March 1998.

[45] D. Hsu, J.C. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. In *Proc. Workshop on Algorithmic Foundations of Robotics (WAFR'00)*, Hanover, NH, March 2000.

[46] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Proceedings of the 1997 IEEE International COnference on Robotics and Automation*, 1997.

[47] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *Int. J. Comp. Geometry and Applications*, 9(4-5):495–512, 1999.

[48] Y.K. Hwang and N. Ahuja. A potential field approach to path planning. *IEEE Transactions of Robotics and Automation*, 8(1):23–32, February 1992.

[49] A. Isidori. *Nonlinear Control Systems.* Springer, 1995.

[50] B. Jakubczyk and E. D. Sontag. Controllability of nonlinear discrete time systems: A Lie-algebraic approach. *SIAM Journal of Control and Optimization*, 28:1–33, 1990.

[51] W. Johnson. *Helicopter Theory.* Princeton University Press, 1980.

[52] T. Kalmár-Nagy, P. Ganguly, and R. D'Andrea. Real-time, near-optimal trajectory control of an omni-directional vehicle. In *Proceedings of the ASME IMECE'01 Symposium on Modeling, Control and Diagnostics of Large-Scale Systems*, 2001.

[53] I. Kaminer, A. Pascoal, E. Hallberg, and C. Silvestre. Trajectory tracking for autonomous vehicles: An integrated approach to guidance and control. *AIAA Journal of Guidance, Control and Dynamics*, 21(1):29–38, January-February 1998.

[54] L. E. Kavraki, M. N Kolountzakis, and J.C. Latombe. Analysis of probabilistic roadmaps for path planning. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, pages 3020–3025, 1996.

[55] L. E Kavraki, J.C. Latombe, R. Motwani, and P. Raghavan. Randomized query processing in robot path planning. *Journal of Computer and System Sciences*, 57(1):50–60, August 1998.

[56] L.E. Kavraki and J.C. Latombe. Probabilistic roadmaps for robot path planning. In K. Gupta and A. del Pobil, editors, *Practical Motion Planning in Robotics: Current Approaches and Future Directions*, pages 33–53. John Wiley, 1998.

[57] L.E. Kavraki, P. Svestka, J.C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[58] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[59] D. E. Koditschek. The application of total energy as a Lyapunov function for mechanical control systems. In J.E. Marsden, P.S. Krishnaprasad, and J.C. Simo, editors, *Dynamics and control of multibody systems*, volume 97 of *AMS*, pages 131–157. Springer-Verlag, New York, NY, 1989.

[60] T.J. Koo, B.Sinopoli, A. Sangiovanni-Vincentelli, and S.S. Sastry. A formal approach to reactive system design: A UAV flight management system design example. In *IEEE International Symposium on Computer-Aided Control System Design*, 1999.

[61] T.J. Koo, F. Hoffmann, B. Sinopoli, and S. S. Sastry. Hybrid control of an autonomous helicopter. In *IFAC Workshop on Motion Control*, 1998.

[62] T.J. Koo and S. Sastry. Output tracking control design of a helicopter model based on approximate linearization. In *Proc. IEEE Conf. on Decision and Control*, December 1998.

[63] W.-S. Koon and J.E. Marsden. Optimal control for holonomic and nonholonomic mechanical systems with symmetry and lagrangian reduction. *SIAM J. of Control and Optimization*, 35(3):901–929, May 1997.

[64] M. Krstic, I. Kanellakopoulos, and P. Kokotovic. *Nonlinear and Adaptive Control Design*. John Wiley & Sons, 1995.

[65] J.J. Kuffner and S.M. LaValle. RRT-Connect: an efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation*, 2000.

[66] G. Laferriere and H. J. Sussmann. A differential geometric approach to motion planning. In Z. Li and J.F. Canny, editors, *Nonholonomic Motion Planning*, pages 235–270. Kluwer, 1993.

[67] J.-C. Latombe. *Robot Motion Planning*. Kluwer, 1991.

[68] J.-C. Latombe. Motion planning: a journey of robots, molecules, digital actors, and other artifacts. *International Journal of Robotics Research*, 18(11):1119–1128, November 1999.

[69] J.-P. Laumond. Singularities and topological aspects in nonholonomic motion planning. In Z. Li and J.F. Canny, editors, *Nonholonomic Motion Planning*, pages 149–200. Kluwer Academic Publishers, Boston, MA, 1993.

[70] J.-P. Laumond, editor. *Robot Motion Planning and Control*, volume 229 of *Lectures Notes in Control and Information Sciences*. Springer, 1998. Available at `http://www.laas.fr/~jpl/book.html` at the time of writing.

[71] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Iowa State University, Ames, IA, Oct. 1998.

[72] S.M. LaValle and J.J Kuffner. Randomized kinodynamic planning. In *Proceedings of the 1999 IEEE International Conference on Robotics and Automation*, 1999.

[73] A. Leonessa, V.S. Chellaboina, and W.M. Haddad. Globally stabilizing controllers for multi-mode axial flow compressors via equilibria-dependent lyapunov functions. In *Proc. IEEE Conf. Dec. Contr.*, San Diego, CA, December 1997.

[74] Z. Li and J.F. Canny, editors. *Nonholonomic Motion Planning.* Kluwer Academic Publishers, Boston, MA, 1993.

[75] J. Liu, X. Liu, T.J. Koo, B. Sinopoli, S.S. Sastry, and E.A. Lee. Hierarchical hybrid system simulation. In *Proc. IEEE Conf. on Decision and Control*, Phoenix, AZ, 1999.

[76] T. Lozano-Pérez. Automaton planning of manipulator transfer movements. *IEEE Transactions on Systems, Man, Cybernetics*, 11(10):681–698, 1981.

[77] T. Lozano-Pérez and M.A. Wesley. An algorithm for planning collision-free paths aming polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[78] D.G. Luenberger. *Optimization by Vector Space Methods.* Wiley, New York, 1969.

[79] J. Lygeros. *Hierarchical Hybrid Control of Large Scale systems.* PhD thesis, University of California, Berkeley, CA, 1996.

[80] J. Lygeros, D. N. Godbole, and S. S. Sastry. Verified hybrid controllers for automated vehicles. *IEEE Trans. Aut. Control*, 43(4):522–539, April 1998.

[81] J. Lygeros, C. Tomlin, and S. Sastry. Controllers for reachability specifications for hybrid systems. *Automatica*, 35(3), March 1999.

[82] N. Lynch, R. Segala, and F. Vandraager. Hybrid I/O automata revisited. In M.D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems IV: Computation and Control*, volume 2034 of *Lecture Notes in Computer Science*, pages 403–417. Springer-Verlag, 2001.

[83] R. Mahony and T. Hamel. Hover control using Lyapunov control function design for an autonomous model helicopter. Submitted to the International Journal of Robotics Research, May 1999.

[84] A. Marigo and A. Bicchi. Steering driftless nonholonomic systems by control quanta. In *Proc. IEEE Conf. on Decision and Control*, 1998.

[85] A. Marigo and A. Bicchi. Rolling bodies with regular surface: Controllability theory and applications. *IEEE Trans. Aut. Control*, 45(9):1586–1599, September 2000.

[86] A. Marigo, B. Piccoli, and A. Bicchi. Reachability analysis for a class of quantized control systems. In *Proc. IEEE Conf. on Decision and Control*, 2000.

[87] J.E. Marsden and T.S. Ratiu. *Introduction to Mechanics and Symmetry*, volume 17 of *Texts in Applied Mathematics.* Springer, 1999.

[88] Ph. Martin. *Contribution à l'étude des systèmes différentiellement plats.* PhD thesis, Ecole Nationale Supérieure des Mines de Paris, 1993.

[89] Ph. Martin. Aircraft control using flatness. In *Multiconference on Computational Engineering in Systems Applications*, pages 194–199, 1996.

[90] Ph. Martin, S. Devasia, and B. Paden. A different look at output tracking: Control of a VTOL aircraft. In *Proc. IEEE Conf. on Decision and Control*, pages 2376–2381, 1994.

[91] N.H. McClamrock and I. Kolmanovsky. Hybrid switched mode control approach for V/STOL flight control problems. In *Proc. IEEE Conf. on Decision and Control*, Kobe, Japan, 1996.

[92] M.W. McConley, B.D. Appleby, M.A. Dahleh, and E. Feron. A computationally efficient Lyapunov-based scheduling procedure for control of nonlinear systems with stability guarantees. *IEEE Transactions on Automatic Control*, January 2000.

[93] K. Mehlhorn and S. N aher. *The LEDA Platform of Combinatorial and Geometric Computing.* Cambridge University Press, 1999.

[94] M. B. Milam, K. Mushambi, and R.M. Murray. A computational approach to real-time trajectory generation for constrained mechanical systems. In *Proc. IEEE Conf. on Decision and Control*, 2000.

[95] S. Monaco and D. Normand-Cyrot. An introduction to motion planning ander multi-rate digital control. In *Proc. IEEE Conf. on Decision and Control*, pages 1780–1785, Tucson, AZ, 1992.

[96] R.M. Murray and S. Sastry. Nonholonomic motion planning: Steering using sinusoids. *IEEE Trans. Aut. Control*, 38:700–716, 1993.

[97] R.M. Murray, Z.Li, and S.S. Sastry. *A Mathematical Introduction to Robotic Manipulation.* CRC Press, 1994.

[98] H. Nijmeijer and A. van der Schaft. *Nonlinear Dynamical Control Systems.* Springer Verlag, New York, NY, 1990.

[99] P.A. O'Dunlaing and C.K. Yap. A retraction method for planning the motion of a disc. *Journal of Algorithms*, 6:104–111, 1982.

[100] K. Oishi and C. Tomlin. Switched nonlinear control of a VSTOL aircraft. In *38th IEEE Conference on Decision and Control*, 1999.

[101] M. H Overmars and P. Svestka. A paradigm for probabilistic path planning. Technical report, Department of Computer Science, Utrecht University, March 1996.

[102] G. D. Padfield. *Helicopter Flight Dynamics: The Theory and Application of Flying Qualities and Simulation Modeling.* AIAA, 1996.

[103] G. J. Pappas, G. Lafferriere, and S. Sastry. Hierarchically consistent control systems. *IEEE Trans. Aut. Control*, 45(6):1144–1160, June 2000.

[104] G.J. Pappas and S. Simic. Consistent hierarchies of nonlinear abstractions. In *Proc. IEEE Conf. on Decision and Control*, pages 4379–4384, Sydney, Australia, 2000.

[105] M. Piedmonte and E. Feron. Aggressive maneuvering of autonomous aerial vehicles: A human-centered approach. In *International Symposium on Robotics Research*, Snowbird, UT, 1999.

[106] L.S. Pontryagin, V.G. Boltanskii, R. Gramkrelidze, and E. Mischenko. *The Mathematical Theory of Optimal Processes.* Interscience Publishers, New York, 1962.

[107] R.W. Prouty. *Helicopter performance, stability, and control.* Krieger Pub. Co., 1990.

[108] J.A. Reeds and R.A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2), 1990.

[109] J.H. Reif. Complexity of the mover's problem and generalizations. In *FOCS*, pages 421–427, 1979.

[110] M. Reyhanoglu, A. van der Schaft, N.H. McClamroch, and I. Kolmanovsky. Dynamics and control of class of underactuated dynamical systems. *IEEE Trans. Aut. Control*, 44(9):1663–1671, 1999.

[111] E. Rimon and D.E. Koditscheck. Exact robot navigation using artificial potential fields. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, 1992.

[112] P. Rouchon, M. Fliess, M. Lèvine, and Ph. Martin. Flatness, motion planning, and trailer systems. In *Proc. IEEE Conf. on Decision and Control*, pages 2700–2705, 1993.

[113] P. Rouchon and J. Rudolph. Invariant tracking and stabilization: problem formulation and examples. In D. Aeyels, F. Lamnabhi-Lagarrigue, and A. van der Schaft, editors, *Stability and Stabilization of Nonlinear Systems*, volume 246 of *Lecture Notes in Control and Information Sciences*, pages 261–273. Springer-Verlag London, London, 1999.

[114] S. Sastry. *Nonlinear Systems: Analysis, Stability, and Control*, volume 10 of *IAM*. Springer Verlag, New York, NY, 1999.

[115] F. Scheck. *Mechanics*. Springer-Verlag, Berlin, Germany, third edition, 1999.

[116] J.T. Schwarz and M. Sharir. On the piano mover's problem: I. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, 36:345–398, 1983.

[117] P. Sedgewick. *Algorithms*. Addison Wesley, 2nd edition, 1988.

[118] C.W. Seibel, J.M. Farines, and J.E.R. Cury. Towards using hybrid automata for the mission planning of unmanned aerial vehicles. In P. Antsaklis, W. Kohn, M. Lemmon, A. Nerode, and S.S. Sastry, editors, *Hybrid Systems V*, volume 1567 of *Lecture Notes in Computer Science*, pages 324–340. Springer-Verlag, Berlin, Germany, 1999.

[119] J.S. Shamma and J.R. Cloutier. Gain-scheduled missile autopilot design using linear parameter varying transformations. *AIAA J. on Guidance, Control an Dynamics*, 16(2):256–263, March-April 1993.

[120] R. L. Shaw. *Fighter Combat: Tactics and Maneuvering*. Naval Institute Press, Annapolis, MD, 1985.

[121] J.M. Shewchun and E. Feron. High performance control with position and rate limited actuators. *International Journal of Robust and Nonlinear Control*, 9(10):617–630, 1999.

[122] Z. Shiller and H. Chang. Trajectory preshaping for high-speed articulated systems. *ASME Journal of Dynamic Systems, Measurement and Control*, 117(3):304–310, 1995.

[123] Z. Shiller, H. Chang, and V. Wong. The practical implementation of time-optimal control for robotic manipulators. *Robotics and Computer-Integrated Manufacturing*, 12(1):29–39, 1996.

[124] R.E. Smith and B.A. Dike. Learning novel fighter combat maneuver rules via genetic algorithms. *International Journal of Expert Systems*, 8(3):247–276, 1995.

[125] E.D. Sontag. A 'universal' construction of Artstein's theorem on nonlinear stabilization. *Systems and Control Letters*, 13(2):117–123, August 1989.

[126] O.J. Sordalen and Y. Nakamura. Design of a nonholonomic manipulator. In *IEEE Conf. on Robotics and Automation*, pages 8–13, 1994.

[127] P. Soueres and J.D. Boissonnat. Optimal trajectories for nonholonomic mobile robots. In J.P. Laumond, editor, *Robot Motion Planning and Control*, volume 229 of *Lecture Notes in Control and Information Sciences*. Springer, 1998.

[128] M.W. Spong. Underactuated mechanical systems. In B. Siciliano and K.P. Valavanis, editors, *Control Problems in Robotics and Automation*. Springer-Verlag, London. UK, 1997.

[129] R. F. Stengel. Toward intelligent flight control. *IEEE Transactions on Systems, Man and Cybernetics*, 23(6):1699–1717, November–December 1993.

[130] H.J. Sussmann. Lie brackets, real analyticity and geometric control. In R. W. Brockett, R. Millman, and H.J.Sussmann, editors, *Differential Geometric Control Theory*, volume 27 of *Progress in Mathematics*, pages 1–116. Birkhauser, 1982.

[131] H.J. Sussmann and V. Jurdjevic. Controllability of nonlinear systems. *Journal of Differential Equations*, 12(1):95–116, July 1972.

[132] D.G. Thomson and R. Bradley. Mathematical definition of helicopter maneuvers. *Journal of the American Helicopter Society*, pages 307–309, October 1997.

[133] M. Tittus and B. Egardt. Control design for integrator hybrid systems. *IEEE Trans. Aut. Control*, 43(4):491–500, April 1998.

[134] C. Tomlin. *Hybrid Control of Air Traffic Management Systems*. PhD thesis, University of California at Berkeley, 1998.

[135] C. Tomlin, J. Lygeros, and S. Sastry. Aerodynamic envelope protection using hybrid control. In *Proc. American Control Conf.*, volume 3, pages 1793–1796, 1998.

[136] C. Tomlin, G. J. Pappas, and S. Sastry. Conflict resolution for air traffic management: A study in multi-agent hybrid systems. *IEEE Trans. Aut. Control*, 43(4), April 1998.

[137] M. J. van Nieuwstadt and R. M. Murray. Real-time trajectory generation for differentially flat systems. *International Journal of Robust and Nonlinear Control*, 8(11):995–1020, September 1998.

[138] M.J. van Nieuwstadt and R.M. Murray. Rapid hover to forward flight transitions for a thrust vectored aircraft. *AIAA J. on Guidance, Control an Dynamics*, 21(1):93–100, Jan-Feb 1998.

[139] G. C. Walsh, R. Montgomery, and S. S. Sastry. Optimal path planning on matrix lie groups. In *Proc. IEEE Conf. on Decision and Control*, 1994.

[140] K. Zhou and J.C. Doyle. *Robust and Optimal Control.* Prentice Hall, 1995.

[141] T. Ziegler. The LEDA guide. Available at: `http://www.algorithmic-solutions.com/leda_guide/Index.html` at the time of writing.