Maneuver-Based Motion Planning for Nonlinear Systems With Symmetries

Emilio Frazzoli, *Associate Member, IEEE*, Munther A. Dahleh, *Fellow, IEEE*, and Eric Feron

***Abstract*—In this paper, we introduce an approach for the efficient solution of motion-planning problems for time-invariant dynamical control systems with symmetries, such as mobile robots and autonomous vehicles, under a variety of differential and algebraic constraints on the state and on the control inputs. Motion plans are described as the concatenation of a number of well-defined motion primitives, selected from a finite library. Rules for the concatenation of primitives are given in the form of a regular language, defined through a finite-state machine called a Maneuver Automaton. We analyze the reachability properties of the language, and present algorithms for the solution of a class of motion-planning problems. In particular, it is shown that the solution of steering problems for nonlinear dynamical systems with symmetries and invariant constraints can be reduced to the solution of a sequence of kinematic inversion problems. A detailed example of the application of the proposed approach to motion planning for a small aerobatic helicopter is presented.**

***Index Terms*—Formal languages, mobile robot motion-planning, optimal control.**

I. Introduction

ONE of the basic problems that have to be solved by mobile robots and autonomous vehicles, and on which this paper will focus, involves computing a motion plan to reach a target from given initial conditions. The motion plan must satisfy a number of differential and algebraic constraints, encoding, for example, nonholonomic constraints on the system's dynamics, actuator position and rate limits, and safety limits on the operational envelope. Moreover, it is often desired that the motion plan make good use of available resources, optimizing a meaningful performance measure.

Perhaps the best-formulated general method for addressing motion-planning problems is the use of optimal control [1]. While general enough to handle, in principle, most motion-planning problems, optimal control techniques suffer from extremely high computational costs, and numerical issues that make them unsuitable for many real-time applications.

Recently, the nonlinear control community has concentrated on analytical methods for steering underactuated controllable mechanical systems in a free environment. (For a thorough review of the state of the art, see [2] and references therein.) However, in many cases of interest, physically accurate models of the dynamics of many vehicles of practical interest do not possess the required properties for direct application of most of the above-mentioned techniques. None of these methods, for example, can be directly applied to realistic aircraft models, subject to nonnegligible aerodynamic forces, actuator saturation, and safety constraints on the state.

On the other hand, expert human pilots are able to effectively operate vehicles with very complicated and possibly unstable dynamics, often at the edge of their operational envelope. In particular, the approach in this paper was motivated by the observation that human pilots execute aerobatic routines through the concatenation of well-practiced "maneuvers" [3]. A formal definition of this concept, and its exploitation in the context of motion planning, are the core of this paper.

A. Languages for Motion Description

A new and promising direction of research in control theory, which will be pursued in this paper, is centered on the purposeful introduction of state and/or control quantization in the design of control systems [4]–[7]. The purpose of quantization is, in general, a reduction of the complexity of the control task. In this paper, instead of quantizing time, the state, or the control input values, we select a finite number of state and control trajectories, which we call motion primitives, and combine them to generate feasible trajectories.

Several approaches to the solution of motion-planning problems have been developed, based on the choice of a finite number of elementary control laws, which are combined to generate more complex behaviors. The approach in this paper is related to other efforts to develop languages and reactive behaviors in robotics; in particular, it can be seen as a structured subclass of the Motion Description Languages in [8]–[11], and a generalization of [5] and [12].

B. Paper Organization

In Section II, we define the class of systems we are interested in, and the kind of problem that we want to solve. In Section III, we discuss the geometric properties of systems with symmetries, introduce the notion of motion primitive, and identify classes of primitives of interest. In Section IV, we define rules for the concatenation of primitives in the form of a regular language: each string in the language corresponds to a feasible trajectory. In Section V, we derive conditions on the language guaranteeing the existence of strings which solve the steering problem. In Section VI, we discuss algorithms to solve a class

Manuscript received December 9, 2004. This paper was recommended for publication by Associate Editor K. Lynch and Editor H. Arai upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation under Grant CCR-0133869.

E. Frazzoli is with the Mechanical and Aerospace Engineering Department, University of California Los Angeles, Los Angeles, CA 90095 USA (e-mail: frazzoli@ucla.edu).

M. A. Dahleh and E. Feron are with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: dahleh@mit.edu; feron@mit.edu).

Digital Object Identifier 10.1109/TRO.2005.852260

of motion-planning problems. In Section VII, we present more in detail the necessary steps for the application of our methodology to a challenging mechanical system, i.e., a realistic model of a small acrobatic helicopter.

II. Problem Formulation

In this paper, we are interested in planning the motion of a time-invariant, nonlinear dynamical control system $\mathcal{S}$, described by a set of ordinary differential equations (ODE) such as

$$\dot{x}(t) := \frac{d}{dt}x(t) = f\left(x(t), u(t)\right) \tag{1}$$

where the state x belongs to an n-dimensional manifold $\mathcal{X}$, and the control u belongs to a set $\mathcal{U} \subseteq \mathbb{R}^m$ (with $m < n$).

Under appropriate technical conditions on the function f (e.g., boundedness and locally Lipschitz dependence on its arguments [2]), and given a piecewise-continuous open-loop control law $\mu : [0, t_{\mathrm{f}}] \to \mathcal{U}$, the ODE (1) can be integrated to compute the state of the system at any time $t \in [0, t_{\mathrm{f}}]$, as a function of the initial conditions, i.e.,

$$x(t) = \varphi_\mu\left(x(0), t\right).$$

The function φ_μ represents the *state flow* of the time-varying dynamical system $\dot{x}(t) = f_\mu(x(t), t) = f(x(t), \mu(t))$, corresponding to (1) under the action of the assigned open-loop control law μ. Given an initial condition $x(0) = x_0$, and a desired final set $\mathcal{F} \subset \mathcal{X}$, the basic *steering problem* can be formulated as the problem of finding a control law μ such that $x(t_{\mathrm{f}}) = \varphi_\mu(x_0, t_{\mathrm{f}}) = x_{\mathrm{f}} \in \mathcal{F}$, for some $t_{\mathrm{f}} \geq 0$.

In practical applications, the motion plan must also satisfy a set of constraints, dictated, for example, by safety considerations and actuator saturation. These conditions can be encoded as a set of inequality constraints on the state and on the control input, of the form

$$F\left(x(t), u(t)\right) \leq 0 \quad \forall t \in [0, t_{\mathrm{f}}] \tag{2}$$

where F is a vector of constraints, and the inequality must be understood component-wise. In the following, we will refer to (2) as *operational envelope* constraints. Typically, these constraints share the invariance properties of the system $\mathcal{S}$, both with respect to time, and with respect to the action of a group. (Invariance properties of $\mathcal{S}$ will be discussed in the next section.) As a measure of the quality of the motion plan, we will consider a cost functional of the form

$$J(x, u) = \int_0^{t_{\mathrm{f}}} \gamma\left(x(t), u(t)\right) dt. \tag{3}$$

As with the constraints, we will assume that the incremental cost function γ shares the same invariance properties of $\mathcal{S}$.

The general motion-planning problem would involve other pointwise-in-time constraints, such as obstacle avoidance, or integral constraints, such as finite fuel. However, dealing with these constraints is beyond the scope of this paper; an application of the methods proposed in this paper to real-time motion planning in dynamic, obstacle-ridden environments can be found in [13].

Finally, we remark that, as is common in the trajectory-generation literature, we will concentrate on open-loop control design. In practical applications, some form of feedback will be needed to handle uncertainties and disturbances in the plant and in the environment. The addition of feedback to the proposed approach is out of the scope of this paper, and will be the subject of future work.

III. Symmetry and Motion Primitives

A fundamental geometric property characterizing the dynamics of many mechanical systems of interest, including most vehicles and moving robots, is invariance with respect to a certain class of transformations on the state of the system.

Consider a finite-dimensional Lie group $\mathcal{G}$, with identity element e. A (left) action of the group $\mathcal{G}$ on the state manifold $\mathcal{X}$ is a smooth map $\Psi : \mathcal{G} \times \mathcal{X} \to \mathcal{X}$, such that $\Psi(e, x) = x$ for all $x \in \mathcal{X}$, and for every $g, h \in \mathcal{G}$, and $x \in \mathcal{X}$, $\Psi(g, \Psi(h, x)) = \Psi(gh, x)$. We will often use the shorthand $\Psi_g(x)$ to mean $\Psi(g, x)$.

We say that a dynamical control system $\mathcal{S}$, described by (1), is *invariant* with respect to the group action Ψ, or, equivalently, that $\mathcal{G}$ is a *symmetry* group for the system $\mathcal{S}$, if for all $g \in \mathcal{G}$, $x_0 \in \mathcal{X}$, $t \in [0, t_f]$, and all piecewise-continuous control laws $\mu : [0, t_f] \to \mathcal{U}$, the following holds:

$$\Psi_g\left(\varphi_\mu(x_0, t)\right) = \varphi_\mu\left(\Psi_g(x_0), t\right).$$

In other words, $\mathcal{G}$ is a symmetry group for $\mathcal{S}$ if its action on the state commutes with the state flow. Invariance also implies that if a curve $t \mapsto (x(t), u(t))$ is an integral of (1), then so is $t \mapsto (\Psi_g(x(t)), u(t))$, for any $g \in \mathcal{G}$. Notice that time invariance of $\mathcal{S}$ implies that the real axis (with addition) is an additional symmetry group, acting on $\mathcal{S}$ by time translation.

A. Motion Primitives

Invariance with respect to a group action leads us to define a notion of equivalence of trajectories, and the concept of motion primitives. We say that two trajectories are equivalent if they can be exactly superimposed through time translation and the action of the symmetry group $\mathcal{G}$. Formally, two (state and control) trajectories $\pi_1 : t \in [t_{\mathrm{i},1}, t_{\mathrm{f},1}] \mapsto (x_1(t), u_1(t))$, and $\pi_2 : t \in [t_{\mathrm{i},2}, t_{\mathrm{f},2}] \mapsto (x_2(t), u_2(t))$, are *equivalent*, if $t_{\mathrm{f},1} - t_{\mathrm{i},1} = t_{\mathrm{f},2} - t_{\mathrm{i},2}$, and if there exist $g \in \mathcal{G}, T \in \mathbb{R}$, such that $(x_1(t), u_1(t)) = (\Psi_g(x_2(t - T)), u_2(t - T))\ \forall t \in [t_{\mathrm{i},1}, t_{\mathrm{f},1}]$.

Consider a time-invariant system $\mathcal{S}$, invariant with respect to actions of the group G, and a trajectory $\pi : [0, t_{\mathrm{f}}] \to \mathcal{X} \times \mathcal{U}$, satisfying (1) and (2). A *Motion Primitive* is the class of trajectories equivalent to π. With a slight abuse of notation, we will also use the symbol π to indicate the corresponding motion primitive, i.e., the set of all trajectories equivalent to π. Let $|\pi|$ indicate the time duration of π. Denote by $\mathcal{P}(\mathcal{S}, \mathcal{G})$ the set of all motion primitives for a time-invariant control system $\mathcal{S}$, with a symmetry group $\mathcal{G}$.

Remark 3.1: Under the assumption that the constraints (2) share the invariance properties of $\mathcal{S}$, i.e.,

$$F(x, u) = F\left(\Psi_g(x), u\right) \quad \forall g \in \mathcal{G}$$

then if a trajectory satisfies the operational envelope constraints, so do all equivalent trajectories. Feasibility with respect to (2) is,

hence, a uniform property of a motion primitive. This assumption is typically verified for actuator limits, and for the operational envelope of the vehicle (e.g., bounds on minimum and maximum speed, etc.).

Remark 3.2: Under a similar assumption on the incremental cost function γ, i.e.,

$$\gamma(x,u) = \gamma\left(\Psi_g(x), u\right) \quad \forall g \in \mathcal{G}$$

all instances of a motion primitive have the same cost. This assumption is satisfied in a broad class of optimal control problems of interest, including minimum-time, minimum-path-length, minimum-energy, and minimum-fuel problems.

In the absence of operational envelope constraints (2), a motion primitive can be obtained simply by applying an arbitrary piecewise-continuous control law μ to the system $\mathcal{S}$, starting from arbitrary initial conditions, for a finite time interval, and storing the resulting state and control trajectory π. This can be computed through integration of (1), or, for example, by running an experiment on a physical system (which gives, by definition, a feasible trajectory). If operational envelope constraints are present, valid trajectories must satisfy these limitations; naturally, experiments need to be limited to such an envelope for safety and other considerations. Note that any fragment of a stored trajectory defined on a compact time interval can be taken as a valid primitive.

B. Concatenation of Motion Primitives

The motion-planning method we present in this paper is based on the selection of a family of motion primitives, which are combined to form complete state and control trajectories satisfying the constraints imposed by the steering problem. We call the operation of combining two primitives to form another primitive *concatenation*. In order to maintain feasibility of trajectories, we cannot concatenate primitives arbitrarily, but must make sure that certain matching conditions are satisfied. More specifically, we need to make sure that the final state of the first primitive coincides with the initial state of the second primitive, modulo an action of $\mathcal{G}$. In the remainder of this section, we give a condition for compatibility of two primitives, and formally define the concatenation operation.

Let us consider two motion primitives $\pi_1 : t \in [0, T_1] \mapsto (x_1(t), u_1(t))$, and $\pi_2 : t \in [0, T_2] \mapsto (x_2(t), u_2(t))$, and let us introduce a *compatibility* relation C between primitives. The primitives π_1 and π_2 are said compatible (written $\pi_1 C \pi_2$), if there exists $g_{12} \in \mathcal{G}$ such that $x_1(T_1) = \Psi(g_{12}, x_2(0))$. Note that C is not symmetric.

If $\pi_1 C \pi_2$, the concatenation of π_1 and π_2 is defined as $\pi_1\pi_2 : [0, T_1 + T_2] \to \mathcal{X} \times \mathcal{U}$, with

$$\pi_1\pi_2(t) = \begin{cases} (x_1(t), u_1(t)), & \text{if } t \le T_1 \\ \left(\Psi\left(g_{12}, x_2(t - T_1)\right), u_2(t - T_1)\right), & \text{otherwise.} \end{cases}$$

Proposition 3.3: The set $\mathcal{P}(\mathcal{S}, \mathcal{G})$ is closed with respect to concatenation of compatible primitives, i.e., if $\pi_1, \pi_2 \in \mathcal{P}(\mathcal{S}, \mathcal{G})$, and $\pi_1 C \pi_2$, then $\pi_1\pi_2 \in \mathcal{P}(\mathcal{S}, \mathcal{G})$.

Proof: Since $\pi_1\pi_2$ coincides with π_1 over $[0, T_1]$, we only need to show that it is continuous at $t = T_1$, and satisfies (1) and (2) over $[T_1, T_1 + T_2]$. Continuity at $t = T_1$ is guaranteed by the compatibility condition

$$\lim_{t \to T_1^-} \pi_1\pi_2(t) = \pi_1(T_1)$$

and

$$\begin{aligned} \lim_{t \to T_1^+} \pi_1\pi_2(t) &= \lim_{t \to T_1^+} \Psi\left(g_{12}, \pi_2(t - T_1)\right) \\ &= \Psi\left(g_{12}, \pi_2(0)\right) \\ &= \pi_1(T_1). \end{aligned}$$

Feasibility for $t \in [T_1, T_1 + T_2]$ is a consequence of invariance to time translation and to the action of $\mathcal{G}$

$$x_{12}(t) = \phi_{\mu_{12}}\left(x_{12}(0), t\right)$$

that is, for $t' = t - T_1 > 0$

$$\begin{aligned} x_{12}(t') &= \varphi_{\mu_2}\left(x_1(T_1), t'\right) \\ &= \varphi_{\mu_2}\left(\Psi\left(g_{12}, x_2(0)\right), t'\right) \\ &= \Psi\left(g_{12}, \varphi_{\mu_2}\left(x_2(0), t'\right)\right) \\ &= \Psi\left(g_{12}, x_2(t')\right). \end{aligned}$$

Going back to $t = T_1 + t'$, the above shows that x_{12} is feasible with respect to (1) and (2) for $t \in [T_1, T_1 + T_2]$, as well. ■

C. Trim Primitives and Maneuvers

In this section, we identify two classes of motion primitives that can be used to build a "library" for motion planning, with certain desirable properties. Intuitively, one can expect that if we select a finite number of primitives, the set of points that can be reached from an initial state, after the concatenation of a finite number of copies of such primitives, will be a discrete set (for a formal analysis of such a system, see [14]).

If a continuous reachable set is desired after the concatenation of a finite number of primitives, one needs to consider a library effectively containing an uncountable number of motion primitives. In order to maintain a finite description for the library, we will look at a particular class of continuously parameterized motion primitives. This class of primitives is identified with steady-state motions, also known as relative equilibria, or trim trajectories in the aeronautical community. Along such a trajectory, controls are kept constant ("trimmed"), and the relative wind has a constant direction with respect to the aircraft.

Formally, we call a nontrivial motion primitive $\alpha : t \in [0, T] \mapsto (x_\alpha(t), u_\alpha(t))$ a *trim primitive* if

$$\begin{cases} x_\alpha(t) = \Psi\left(\exp(\xi_\alpha t), x_\alpha(0)\right) \\ u_\alpha(t) = u_\alpha \end{cases} \quad \forall t \in [0, T] \tag{4}$$

where ξ_α is an element of $\mathfrak{g}$, the Lie algebra of $\mathcal{G}$. In other words, trim primitives correspond to finite flows along left-invariant vector fields. We indicate the set of all trim primitives for the system $\mathcal{S}$ with symmetry group $\mathcal{G}$ as $\mathcal{T}(\mathcal{S}, \mathcal{G})$. The simplest example of a trim primitive is an equilibrium point.

Given a trim primitive α, one can build a whole family of trim primitives parameterized by a scalar $\tau \ge 0$ by extending or restricting its domain of definition, as follows:

$$\alpha(\tau) : t \in [0, \tau] \mapsto \left(\Psi\left(\exp\left(\xi_\alpha t, x_\alpha(0)\right)\right), u_\alpha\right). \tag{5}$$

Let $\mathcal{T}_\alpha = \{\alpha(\tau), \tau \geq 0\} \subset \mathcal{T}(\mathcal{S},\mathcal{G})$ be the set of all trim primitives obtained by changing the domain of a trim primitive α. We call the nonnegative scalar τ the *coasting time*, since it determines how much time is spent executing a trim primitive, and how long the system follows the flow of the corresponding left-invariant vector field.

Finally, we define a *maneuver* as a nontrivial primitive which is compatible, from the left and from the right, with trim primitives; in other words, we define a maneuver as a primitive that begins and ends at steady-state conditions. If we denote by $\mathcal{M}(\mathcal{S},\mathcal{G}) \subseteq P(S,G)$ the set of all maneuvers, we have that $\pi \in \mathcal{M}(\mathcal{S},\mathcal{G}) \Leftrightarrow \exists \alpha, \beta \in \mathcal{T}(\mathcal{S},\mathcal{G}) : \alpha\pi\beta \in \mathcal{P}(\mathcal{S},\mathcal{G})$. We call the set of trim primitives that are compatible with π from the left the *predecessors* of π, and indicate it as $\mathrm{Pred}(\pi) \subseteq \mathcal{T}(\mathcal{S},\mathcal{G})$. Correspondingly, we define, as $\mathrm{Succ}(\pi)$, *successors* of π, the set of trim primitives that are compatible with π from the right. Notice that, if $\alpha \in \mathrm{Pred}(\pi)$ (resp. $\beta \in \mathrm{Succ}(\pi)$), then $\mathrm{Pred}(\pi) = \mathcal{T}_\alpha(\mathcal{S},\mathcal{G})$ (resp. $\mathrm{Succ}(\pi) = \mathcal{T}_\beta(\mathcal{S},\mathcal{G})$).

Consider a maneuver $\pi \in \mathcal{M}(\mathcal{S},\mathcal{G})$, of duration T_π. Since, by definition, the maneuver π is compatible with a trim primitive α from the left, and with a trim primitive β from the right, there must be $g_{\alpha\pi}, g_{\pi\beta} \in \mathcal{G}$ such that $x_{\pi(0)} = \Psi(g_{\alpha\pi}, x_\alpha(0))$, and $x_\pi(T_\pi) = \Psi(g_{\pi\beta}, x_\beta(0))$. We define the *group displacement* of the maneuver π as $g_\pi = g_{\alpha\pi}^{-1} g_{\pi\beta}$.

Remark 3.4: While $g_{\alpha\pi}$ and $g_{\pi\beta}$ depend on the choice of particular class representatives, the group displacement g_π is $\mathcal{G}$-invariant, i.e., it is an invariant characteristic of a maneuver.

IV. The Maneuver Automaton

The proposed method for motion planning relies on the choice of a finite set of maneuvers $\Sigma \subset \mathcal{M}(\mathcal{S},\mathcal{G})$, and the generation of complex trajectories through the concatenation of maneuvers in Σ. Since not all maneuvers are compatible, rules for the choice of "legal" sequences are needed. A convenient way of representing such rules is given by the definition of a formal language. In other words, we will consider the set Σ as the alphabet of a formal language. Words in this language are formed through concatenation of several maneuvers, hence, a word will be an element of the so-called *free monoid* Σ^*, i.e., the set of all possible sequences of symbols in Σ. (The identity element is the null string ϵ.) In general, not all strings in Σ^* correspond to feasible trajectories with respect to (1), since only compatible primitives can be concatenated. It is convenient to represent all strings in Σ^* formed by concatenation of compatible primitives as the set of all strings accepted by a finite state machine, which we will call a *Maneuver Automaton* (MA). An MA is a tuple

$$\mathrm{MA} = \{\Sigma, Q, \delta, q_0, F\} \tag{6}$$

where we have the following.

- $\Sigma \subset \mathcal{M}(\mathcal{S},\mathcal{G})$ is the maneuver alphabet, i.e., a finite collection of maneuvers.
- $Q = Q_0 \cup \{\square\}$ is a finite set of states. The set $Q_0 \subset \mathcal{T}(\mathcal{S},\mathcal{G})$ is a collection of trim primitive closures chosen in such a way that for any $\pi \in \Sigma$, $\mathrm{Pred}(\pi), \mathrm{Succ}(\pi) \in Q_0$, and $\cup_{\pi\in\Sigma}\{\mathrm{Pred}(\pi), \mathrm{Succ}(\pi)\} = Q_0$. In other words, the set Q_0 is a minimal set that includes all trim primitives from which maneuvers can start, and at which maneuvers can end. The symbol $\square$ represents an error state, used to detect unallowed strings.
- The map $\delta : Q \times \Sigma \to Q$ is a transition function, relating the state after the execution of a maneuver, to the state before the maneuver. The map δ is defined as follows:

$$\delta(\alpha,\pi) = \begin{cases} \beta \in Q : \beta \in \mathrm{Succ}(\pi), & \text{if } \alpha \in \mathrm{Pred}(\pi) \\ \square, & \text{otherwise.} \end{cases}$$

- $q_0 \in Q$ is an initial state.
- $F \subset Q$ is a set of final, or accepting, states.

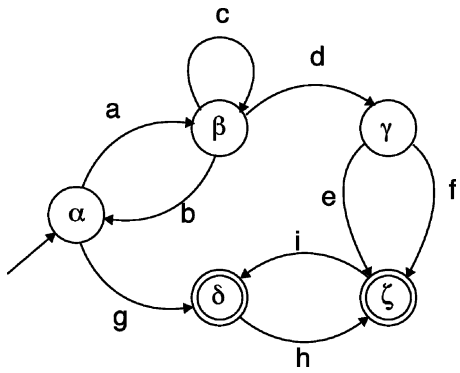

Fig. 1. Example of a digraph representing an MA.

An MA can be conveniently depicted as a directed graph in which vertices represent states (i.e., trim primitives), and edges represent maneuvers. The initial state is indicated by an arrow, and final states are indicated with a double circle; see Fig. 1 for an example. In the figure, trim primitives are labeled with Greek letters, and maneuvers are labeled with Roman letters. An MA graph need not be connected or strongly connected, admits parallel edges (i.e., more than one maneuver can connect the same trim primitives, as in the case of `e` and `f` in the figure), and edges having the same source and target node, such as `c` in the figure.

Given an MA, we define the language $L(\mathrm{MA}) \subseteq \Sigma^*$ as the set of all strings accepted by it. We will call a string $\omega \in L(\mathrm{MA})$ a *maneuver sequence*. Since an MA is a finite state machine, the language $L(\mathrm{MA})$ is a regular language. Other ways to define regular languages include regular grammars. Production rules to obtain all strings in $L(\mathrm{MA})$ can be derived from the compatibility relation; we will not pursue this straightforward task.

We will make the assumption that the initial and final conditions for the motion-planning problem are such that they can be represented as trim primitives in Q, i.e., there exist $g_0 \in \mathcal{G}$ and $\alpha \in Q$ such that $x_0 = \Psi_{g_0}(x_\alpha(0))$. Similar assumptions are made for the target set, i.e., $\forall x_f \in \mathcal{F}, \exists (g_f, \beta) \in \mathcal{G} \times \mathcal{F} : x_f = \Psi_{g_f}(x_\beta(0))$. These assumptions are usually verified in motion planning, since initial and final conditions are typically defined at rest, or at some steady-state condition. In (6), Q, Σ, and δ depend on the system, and on the choice of primitives to include in the language. On the other hand, q_0 and F depend on the specific instance of the problem to be solved. When discussing properties that should hold uniformly over any choice of q_0 and F, such as controllability, we will leave these undefined.

A. Motion Plans

The role of trim primitives has been, so far, just that of providing a convenient way of formally defining a special kind of motion primitives, namely, maneuvers, and providing them with a common interface for concatenation. However, trim primitives can be further exploited to enrich trajectories that can be represented using the MA language.

Consider the maneuver sequence $\omega = \pi_1, \pi_2, \ldots, \pi_N$; by inserting zero-length trim primitives between maneuvers, the primitive ω can be equivalently represented as $\omega = \alpha_1(0)\pi_1, \alpha_2(0)\pi_2, \ldots \pi_N\alpha_{N+1}(0)$, with $\alpha_i \in \mathrm{Pred}(\pi_i)$ for $i = 1, \ldots, N$, and $\alpha_{N+1} \in \mathrm{Succ}(\pi_N)$. We can obtain a much richer set of primitives by interleaving maneuvers with nonnegative flows along trim primitives. In other words, we can consider primitives of the form $(\omega, \tau) = \alpha_1(\tau_1)\pi_1, \alpha_2(\tau_2)\pi_2, \ldots \pi_N\alpha_{N+1}(\tau_{N+1})$, $\tau_i \geq 0\ \forall i \in \{1, N+1\}$. As in the previous case, $(\omega, \tau) \in \mathcal{P}(\mathcal{S}, \mathcal{G})$ by construction.

Based on this discussion, we define a *motion plan* on an MA as a pair $(\omega, \tau) \in L(\mathrm{MA}) \times \bar{\mathbb{R}}_+^{N(\omega)+1}$, where $N(\omega)$ is the number of symbols in ω, and τ is an array of $N(\omega)+1$ nonnegative coasting times.

While a maneuver sequence can be associated with a path on a directed graph representing the MA, a motion plan augments this information by specifying the time spent at the states (trim primitives). The duration of the motion plan (ω, τ) is equal to the sum of the duration of the maneuver sequence ω and the sum of the coasting times in the array τ.

A key property of our language for the purpose of motion planning is the ability to recover in a mathematically convenient way the complete state of the system at any time during the execution of a motion plan, without resorting to simulation or numerical integration of the differential equations (1). To see this, consider a motion plan of length one, i.e., a motion plan composed of just one maneuver, preceded and followed by a trim primitive. The motion plan is defined by the primitive $(\omega, \tau) = (\pi_1, [\tau_1, \tau_2]) = \alpha_1(\tau_1)\pi_1\alpha_2(\tau_2)$. If $x(0) = \Psi_{g_0}(x_{\alpha_1}(0))$, the state trajectory is described by

$$x(t) = \begin{cases} \Psi(g_0 \exp(\xi_{\alpha_1} t), x_{\alpha_1}(0)), & t \in [0, \tau_1] \\ \Psi(g_0 \exp(\xi_{\alpha_1}\tau_1) g_{\alpha_1\pi_1}, \pi_1(t-\tau_1)), & t \in [\tau_1, \tau_1 + |\pi_1|) \\ \Psi(g_0 \exp(\xi_{\alpha_1}\tau_1) g_{\pi_1} \\ \quad \cdot \exp(\xi_{\alpha_2}(t - \tau_1 - |\pi_1|)), x_{\alpha_2}(0)), & \text{otherwise.} \end{cases}$$

The above formula can be easily extended to a motion plan of arbitrary length. In particular, the final state after the execution of a motion plan $p = (\omega, \tau)$ of length $N(\omega)$ is given by $x_{\mathrm{f}} = \Psi(g_0 g_p, x_{\alpha_{N(\omega)+1}}(0))$, where the *group displacement* g_p can be computed as

$$g_p = \left[\prod_{i=1}^{N(\omega)} \exp(\xi_{\alpha_i}\tau_i)\, g_{\pi_i} \right] \exp\left(\xi_{\alpha_{N(\omega)+1}} \tau_{N(\omega)+1}\right). \quad (7)$$

In the above, we used the convention that π_i is the ith symbol in the maneuver sequence ω, $\alpha_i \in \mathrm{Pred}(\pi_i)$, $i = 1, \ldots, N(\omega)$, and $\alpha_{N(\omega)+1} \in \mathrm{Succ}(\pi_{N(\omega)})$. Moreover, successive factors in the product between square brackets must be right-multiplied. Equation (7) can be rewritten in the equivalent form

$$g_p = g_\omega \prod_{i=1}^{N(\omega)+1} \exp(\eta_i \tau_i) \quad (8)$$

where g_ω is the group displacement corresponding to the motion plan $(\omega, 0)$, and the vector fields η_i are defined by

$$\xi_{\alpha_i} = \mathrm{Ad}\left(\prod_{j=i}^{N(\omega)} g_{\pi_j}, \eta_i \right) \quad (9)$$

and $\eta_{N(\omega)+1} = \xi_{\alpha_{N(\omega)+1}}$; $\mathrm{Ad}(g, \cdot)$ is the adjoint mapping of $g \in \mathcal{G}$, i.e., $\mathrm{Ad}(g, \xi) = g\xi g^{-1}$.

Equation (8) shows that the total group dispacement due to a motion plan p can be decomposed into a group displacement g_ω, due to the maneuver sequence ω, and the composition of flows along left-invariant vector fields η, which can be computed from the maneuver sequence. Remarkably, (8) has the structure of a *forward kinematic map* in robotics, such as, for example, the map defining the position of the end-effector of a robotic arm, as a function of the joint angles [15]. In other words, the nonlinear dynamical system (1) can be transformed, using the MA language and formalism, into a system that formally behaves like a kinematic system. The transformation is exact, i.e., no approximations are introduced, but it limits the admissible trajectories to sequential combinations of a finite number of motion primitives, computed *a priori*.

Kinematic decoupling, introduced in [16], also leads to the reformulation of steering problems for dynamical systems as kinematic inversion problems. However, kinematic decoupling is applicable only to systems with symmetries satisfying certain additional conditions, and requires the system to stop each time motion along a different vector field is desired. Our approach does not suffer from these limitations.

B. Maneuver Automata and Other Languages for Motion Description

Some remarks about the relationship of the language defined in this section to similar efforts in the literature are appropriate at this point. Languages defined via Maneuver Automata can be seen as a subset of motion-description languages and extensions, such as MDLe, as defined in [8] and [9]. The elementary component of MDLe is an *atom*, composed of a control law, a set of Boolean triggers acting on sensory information, and a timer. Both the triggers and the timer govern transition between atoms. Since we focus on controllability analysis and feasible trajectory generation, as opposed to reactive behavior under external stimuli, we do not model the Boolean triggers. Each primitive in a motion plan can be seen as a triggerless atom, i.e., a control law, initiated at some specified initial conditions, together with a timer. If the motion primitive is a maneuver, the timer is set to its duration; if it is a trim primitive, the timer is set to the corresponding coasting time.

The additional structure provided by choosing the symbols in the language as equivalence classes under group actions, and

further restricting the choice of symbols to well-defined maneuvers, makes it possible to explicitly take symmetries into account, and exploit them to simplify the representation and computation of trajectories. For example, in our language, it is possible to write down explicitly the final state after the execution plan (8); in general, this is not possible for an arbitrary motion-description language, without resorting to numerical integration of (1).

Maneuver Automata can be seen as a generalization of other motion-planning techniques, respectively based on *control quanta* and *motion graphs*. Control quanta were originally conceived as a motion-planning method for driftless systems with symmetries [5]. Each control quantum is a finite-time open-loop control law; at the end of the execution of a control quantum, the control input is set to zero, hence bringing the system to a stop. In our language, each control quantum is a maneuver, starting and ending at an equilibrium, i.e., a zero-velocity trim primitive. Maneuvers, according to our definition, differ from control quanta in the sense that they are not constrained to start and end at zero-velocity conditions. As a consequence, the MA formalism allows us to deal in a natural way with systems with drift, and allows the system to flow along nontrivial vector fields, with important consequences on reachability properties.

Motion graphs were introduced to build complex animations of computer-generated characters from motion-capture sequences [12]. Given a finite number of motion primitives, a motion graph can be constructed by representing rules for their sequential combination as a directed graph. Motion primitives are constructed from motion-capture sequences by identifying frames at which different segments can be joined. (In computer animation, one is not constrained by the continuity of the state and control trajectories, as long as the viewer does not perceive the discontinuities; hence, the conditions for concatenation are relaxed.) In essence, motion graphs are constructed as Maneuver Automata, using general motion primitives, instead of maneuvers. (In other words, without imposing the constraint that motion primitives start and end at common trim primitives.)

The key advantage of Maneuver Automata is that our formulation adds the ability to handle flows along nontrivial left-invariant vector fields to both the control quanta and motion graph concepts. The main implications of this added capability are addressed in the following sections.

V. Controllability of Maneuver Automata

Our approach to the solution of motion-planning problems can be summarized as the following. Instead of looking for a control input in an infinite-dimensional set (e.g., piecewise-continuous functions), we limit our search to motion plans defined on the regular language $L(\mathrm{MA})$. By limiting our search to plans composed of a finite number of primitives and coasting times, we reformulate a differential problem, such as the steering problem, as a finite-dimensional algebraic problem.

On the other hand, these computational advantages come at the cost of reducing the set of achievable trajectories for $\mathcal{S}$. The main issue at this point is the choice of the finite set of primitives to include in the maneuver library Σ. A key requirement on Σ, and on the resulting MA, is the preservation of reachability properties of the original system $\mathcal{S}$. Since we limit the initial and final conditions to be trim primitives, the reachable set from any initial state cannot include the entirety of the set $\mathcal{X}$. On the other hand, the MA language has enough expressive power to make it possible, under certain conditions to be derived, to encode in this language a motion plan to reach any point in the orbit of x_q, $q \in F$ under the action of $\mathcal{G}$.

We say that an automaton MA is *controllable* (uniformly on q_0 and F) if, for any initial condition $x(0) = \Psi(g_0, x_{q_0}(0))$, any final trim primitive $q_\mathrm{f} \in F$, and any compact set $\mathcal{H} \subseteq \mathcal{G}$, there exists a time T which is an upper bound on the time necessary to reach a desired target $x_\mathrm{f} = \Psi(g_\mathrm{f}, x_{q_\mathrm{f}}(0))$, with $g_\mathrm{f} \in H$. Because of the invariance to actions of $\mathcal{G}$, controllability is a global property.

We define a *fixed-point* motion plan as a pair $p_0 = (\omega_0, \tau_0)$, where ω_0 is a word describing a closed path on the MA, and τ_0 is chosen in such a way that $x(|p_0|) = x(0)$, i.e., the group displacement caused by the motion plan p_0 is equal to the identity $g_{p_0} = e$. We want to analyze what happens if a fixed-point motion plan is perturbed by adding a vector $\delta\tau$ to the nominal coasting times τ_0. In other words, we are interested in studying the map $\Delta_{p_0} : \mathbb{R}^{N+1} \to \mathcal{G}$, $\delta\tau \mapsto g_{(\omega_0, \tau_0 + \delta\tau)}$. (Obviously, $\Delta_{p_0}(0) = e$, by definition of fixed-point motion plan.) Given a scalar $\theta \leq \|\tau_0\|_\infty$, define the set $\mathcal{R}_{p_0}(\theta) := \{g \in \mathcal{G} : \exists \delta\tau, \|\delta\tau\|_1 \leq \theta, \Delta_{p_0}(\delta\tau) = g\}$, i.e., the set of points which can be reached within time $|p_0| + \theta$, by perturbing the coasting times τ.

Theorem 5.1 (MA Controllability): An MA is controllable, uniformly on q_0 and F, if and only if:

(1) for any $\alpha_0, \alpha_\mathrm{f} \in Q$, and for any pair $\mathcal{G}_i, \mathcal{G}_j$ of connected components of $\mathcal{G}$, there is a maneuver sequence $\omega \in L(\mathrm{MA}(Q, \Sigma, \delta, \alpha_0, \{\alpha_\mathrm{f}\}))$ such that $g_i g_\omega \in \mathcal{G}_j \ \forall g_i \in \mathcal{G}_i$. In other words, the directed graph representing the MA is strongly connected, and the symmetry group $\mathcal{G}$ can be connected through the action of maneuvers;

(2) there exists a finite-time, fixed-point motion plan $p_0 = (\omega_0, \tau_0)$, with $\|\tau_0\|_\infty > 0$, such that

$$e \in \operatorname{int} \mathcal{R}_{p_0}(\theta) \quad \forall \theta > 0. \tag{10}$$

Proof: **Necessity**: The necessity of the first condition is a direct consequence of the definition of controllability; if it were not satisfied, no path would exist between initial and final conditions for some choice of trim primitives, and in arbitrary connected components of $\mathcal{G}$. The necessity of the existence of a fixed-point maneuver sequence is proven by contradiction, by breaking the problem into two steps, i.e., proving the necessity of fixed-point motion plans, and then the necessity of (10). Assume that the system is controllable, but there are no fixed-point motion plans. Since the system is controllable, it is possible to find a motion plan p_1 to steer it from $x_0 = x(0)$ to $x_1 = \Psi_{g_\mathrm{f}}(x_{\alpha_\mathrm{f}})$ in finite time, for any $g_\mathrm{f} \in H$. After the first step is done, it is possible to find another motion plan p_2 to steer it from x_1 to x_0, again in finite time. The composition of the maneuver sequences $\bar{v}_1$ and $\bar{v}_2$ is a fixed-point maneuver, which contradicts the assumption.

The next point is proving that there must exist a *single*, finite-time, fixed-point motion plan $(\tilde{\omega}, \tilde{\tau})$ such that the corresponding reachable set under coasting-time perturbations

satisfies (10). 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 nonempty interior. Consider a compact set $\mathcal{H} \subseteq \mathcal{G}$ with a nonempty interior. Since the system is controllable, it will be possible to find a finite time T such that any point in $\mathcal{H}$ will be reachable through at least one motion plan p_1 of duration $T_1 \leq T$. Because of controllability, for any motion plan p_1, it will be possible to find a second motion plan p_2 such that the sequential combination of the two results into a fixed-point plan. Since we assumed that there is no fixed-point sequence with a reachable set with a nonempty interior, and any motion plan maps open sets into open sets, we must conclude that the reachable set of any motion plan p_1 has an empty interior. The number of all the possible maneuver sequences ω which result in a total sequence duration smaller that T 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 $\mathcal{H}$ (a set with a nonempty interior) with a finite number of sets with empty interiors.

Sufficiency: Assume that there is a fixed-point motion plan p_0, satisfying (10). Because of the first condition in the theorem, it is possible to find a motion plan p_1 that takes the system from the initial trim primitive α_i to a trim primitive α_0, compatible with one of the maneuvers in ω_0; the motion plan will result in a group displacement g_{p_1}. Similarly, it is possible to find a motion plan p_2 that takes the system from the trim primitive α_0 to the final trim primitive α_j; such a motion plan results in a group displacement g_{p_2}.

The problem of finding a finite motion plan to steer the system from the initial and final conditions is reduced to finding a motion plan steering the system from $\Psi(g_{p_1}, x_{\alpha_0}(0))$ to $\Psi(g_{p_2}^{-1}, x_{\alpha_0}(0))$. Because of the first condition in the theorem, it is always possible to construct p_1 and p_2 in such a way that $h_1 = g_0 g_{p_1}$ and $h_2 = g_f g_{p_2}^{-1}$ are in the same connected component of $\mathcal{G}$. Consider a continuous curve $\eta : [0,1] \to \mathcal{G}$, such that $\eta(0) = e$ and $\eta(1) = h_1^{-1} h_2$. The second condition, together with invariance with respect to time and to actions of $\mathcal{G}$, ensures that it is always possible to find $\varepsilon > 0$ in such a way that $\eta(t)^{-1}\eta(t+\varepsilon) \in \mathcal{R}_{p_0}(\|\tau_0\|_\infty)$, for all $t \in [0, 1-\varepsilon]$. In other words, it is possible to move along the curve η by at least ε through a motion plan of the form $(\omega_0, \tau_0 + \delta\tau)$; as a consequence, with at most $1/\varepsilon$ steps, one can reach the target point exactly. The total duration of a motion plan to reach a set with an open interior containing the target point can thus be bounded by $T \leq |p_1| + |p_2| + 1/\varepsilon(|p_0| + \|\tau_0\|_1)$. Since any compact set $\mathcal{H} \subseteq \mathcal{G}$ can be covered by a finite number of sets built as detailed above, it is possible to find a finite bound on the time to reach any point in $\mathcal{H}$. ∎

Given a fixed-point motion plan $p_0 = (\omega_0, \tau_0)$, we are interested in computational tests to ensure that the corresponding reachable set has a nonempty interior. In the limit $\|\delta\tau\| \to 0$, the composition of flows along the vector fields identified by the vectors $\eta_{0,i}$, defined by

$$\xi_{\alpha_i} = \mathrm{Ad}\left(\prod_{j=1}^{N(\omega)} g_{\pi_j} \exp\left(\xi_{\alpha_j} \tau_{0,j}\right), \eta_{0,i} \right) \tag{11}$$

is equivalent to the flow along a linear combination of the same vectors. In other words, as $\|\delta\tau\| \to 0$, $\Delta_{p_0}(\delta\tau) = \prod_i \exp(\eta_{0,i}\delta\tau_i) \approx \exp(\sum_i \eta_{0,i}\delta\tau_i)$. Clearly, the set $\mathcal{R}_{p_0}(\theta)$ has a nonempty interior if the distribution

$$\Xi = \{\eta_{0,1}, \eta_{0,2}, \ldots, \eta_{0,N+1}\}$$

spans the Lie algebra $\mathfrak{g}$. Even in the case in which p_0 does not satisfy this condition, it could do so if repeated a sufficiently large number of times. Indicate the motion plan obtained through sequential combination of L copies of p_0 as p_0^L. A sufficient condition for controllability of an MA is then the following version of the Lie algebra rank condition [17].

Theorem 5.2: The reachable set $\mathcal{R}_{p_0^L}(\theta)$ has a nonempty interior for all $\theta > 0$ and some $L \in \mathbb{N}$, if $\|\tau_0\|_\infty > 0$ and the involutive closure $\bar{\Xi}$, under the Lie bracket operation, of the set Ξ has the same dimension as the Lie algebra $\mathfrak{g}$.

Proof: The system $g[i+1] = g[i]\Delta_{p_0}(\delta\tau)$, with $g[0] = e$, is an invertible, locally (in the "controls" $\delta\tau$) analytic nonlinear discrete-time system. The result follows from the application of [18, Th. 3 and 9]. ∎

Note that this is a nontrivial result, because the Lie algebra rank condition cannot be applied directly to the vector fields ξ_{α_i}, since only nonnegative flows are allowed along them.

The above theorems show that the addition of trim primitives, i.e., the ability to move along left-invariant vector fields on $\mathcal{G}$, provides the "slackness" necessary to achieve a continuous reachable set using a finite number of motion primitives. The case in which there is no such capability has been analyzed in [14], where it is shown that the reachable set has the structure of a lattice, which can be either dense everywhere, or nowhere. A continuous reachable set results in bounded times to reach exactly any point in a compact set in $\mathcal{G}$; whereas in the case of a discrete reachable set, it is not possible to bound the time needed to get arbitrarily close to some elements of $\mathcal{G}$.

Controllability of an MA implies neither small-time nor local controllability of the underlying dynamical system $\mathcal{S}$, but it does imply a form of configuration controllability, where the configuration is restricted to group variables. Conversely, local controllability implies that it is possible to define a set of maneuvers rich enough to yield a controllable MA. The proof of the controllability theorem implies the knowledge of a fixed-point motion plan; simple rules for constructing such motion plans for arbitrary systems, invariant to actions of $SE(2)$ (e.g., car-like robots) and of $SE(2) \times (\mathbb{R}, +)$ (e.g., aircraft) are given in [19], and are not reported here due to space limitations. See Section VII for an example.

VI. Maneuver-Based Motion Planning

Given a fixed-point motion plan p_0, the proof of *Theorem 5.1* provides a procedure to construct a motion plan satisfying the boundary conditions and the envelope constraints discussed in Section II. However, such a motion plan will be, in general, very inefficient; the objective of this section is to present a method to compute motion plans that minimize a cost of the form (3). Using Maneuver Automata, it is possible to *approximate* solutions to a class of infinite-dimensional differential problems solving finite-dimensional algebraic problems. On the

other hand, restricting trajectories to combinations of a finite number of motion primitives results in the addition of additional constraints: hence, the solutions computed according to the proposed method will not, in general, be optimal. The relationship between initial and final condition in a motion plan (ω, τ) is encoded by (8). Moreover, any motion plan resulting from the concatenation of feasible motion primitives satisfies the operational envelope constraints, see *Remark 3.1*. It remains to specify the cost of a motion plan.

Proposition 6.1: The cost (3) of a motion plan $p = (\omega, \tau)$ can be written as

$$J(x_p, u_p) = J_{\mathrm{MA}}(p) = \sum_{i=1}^{N} (\Gamma_{\pi_i} + \gamma_{\alpha_i} \tau_i) + \gamma_{\alpha_{N+1}} \tau_{N+1} \tag{12}$$

where Γ_π is the cost of the motion primitive π, and γ_α represents the cost per unit time of the trim primitive α, i.e., the cost of $\alpha(1)$.

Proof: Split the motion plan p into two parts, such that $p = p_1 p_2$. Since the cost (3) is additive and time-invariant

$$\begin{aligned} J(x_p, u_p) &= J(x_{p_1}, u_{p_1}) + J(x_{p_2}, u_{p_2}) \\ &= \int_0^{|p_1|} \gamma(x_{p_1}(t), u_{p_1}(t))\, dt \\ &\quad + \int_0^{|p_2|} \gamma(x_{p_2}(t), u_{p_2}(t))\, dt. \end{aligned}$$

Similarly, it is possible to split the cost of a motion plan p into the sum of the costs of each individual motion primitive.

As noted in *Remark 3.2*, the cost of a motion primitive is a primitive-specific constant, i.e., it does not depend on the time or state at which the primitive is initiated. Furthermore, the cost of a trim primitive depends linearly on its duration

$$\begin{aligned} J(x_\alpha, u_\alpha) &= \int_0^{\tau} \gamma(x_\alpha(t), u_\alpha(t))\, dt \\ &= \int_0^{\tau} \gamma(\Psi(g_0 \exp(\xi_\alpha t), x_\alpha(0)), u_\alpha)\, dt \\ &= \int_0^{\tau} \gamma(x_\alpha(0), u_\alpha)\, dt \\ &= \gamma_\alpha \tau. \end{aligned}$$

Hence, (12) corresponds to the breakdown of the cost of a motion plan into the sum of the primitives from which it is constructed. ■

Remark 6.2: The cost of a motion plan $p = (\omega, \tau)$ is affine in the coasting times τ.

Given a library of motion primitives defining a language MA, the most efficient motion plan solving a steering problem can be found by solving the following nonlinear program:

$$\begin{aligned} (\omega^*, \tau^*) = \arg\min_{(\omega,\tau)} & J_{\mathrm{MA}}(\omega, \tau) \\ \text{s.t.} : g_\omega & \prod_{i=1}^{N(\omega)+1} \exp(\eta_i \tau_i) = g_0^{-1} g_f \\ & \omega \in L(\mathrm{MA}) \\ & \tau_i \geq 0 \quad \forall i = 1, \ldots, N(\omega) + 1. \end{aligned} \tag{13}$$

Before proceeding further, we must first ensure that an optimal solution to (13) does indeed exist.

Theorem 6.3: If the MA is controllable, then there exists a solution of the optimal control problem (13). The optimal motion plan will have finite length and finite cost.

Proof: Since the MA is controllable, there exists a finite-length motion plan $\bar{p} = (\bar{\omega}, \bar{\tau})$ which satisfies the boundary conditions, with finite cost. Since the cost of any nonempty motion plan is at least $\varepsilon = \min_{\pi \in \Sigma} \Gamma_\pi > 0$, there is a finite number of maneuver sequences in $L(\mathrm{MA})$, encoding motion plans with a cost possibly smaller than $J_{\mathrm{MA}}(\bar{\omega}, \bar{\tau})$, e.g., the set of all maneuver sequences of length bounded by $J_{\mathrm{MA}}(\bar{p})/\varepsilon$. For each of these maneuver sequences, i.e., for a fixed ω, program (13), with the additional constraint $J_{\mathrm{MA}}(p) \leq J_{\mathrm{MA}}(\bar{p})$, is a smooth optimization problem over a compact domain. Such a problem will either have an (attained) optimal solution, or be unfeasible. Hence, in addition to the solution candidate $\bar{\omega}$ obtained from the controllability theorem, there will be a finite number of other candidates for an optimal solution. The candidates with the smallest cost are optimal solutions to (13). The optimal solution is not necessarily unique. ■

The optimization problem posed in (13) is, in general, nonconvex and difficult to solve; however, since it is a finite-dimensional problem, its solution is, in general, easier than the solution of its differential counterpart. The structure of the group $\mathcal{G}$, and of the trim primitives included in the set of Automaton states Q_0, can induce further simplifications. For example, we can state the following.

Proposition 6.4: Consider an MA $\mathrm{MA}(\Sigma, Q, \delta, q_0, F)$, such that the set $Q_0 = Q \backslash \{\square\}$ is such that $\xi_\alpha \in \mathfrak{se}(2)$ for all $\alpha \in Q_0$. Under these assumptions, the program (13), fixed as maneuver sequence ω, is a *polynomial program* in the coasting times τ.

Proposition 6.5: Consider an MA $\mathrm{MA}(\Sigma, Q, \delta, q_0, F)$, such that the set $Q_0 = Q \setminus \{\square\}$ is such that $\xi_\alpha \in \mathbb{R}^n$ for all $\alpha \in Q_0$. Under these assumptions, the program (13), fixed as maneuver sequence ω, is a *linear program* in the coasting times τ.

The structure of (13) lends itself to a hierarchical decomposition of the search for the optimal motion plan into a combinatorial problem, i.e., the choice of the optimal maneuver sequence ω^*, and a smooth, generally nonconvex optimization problem, i.e., the determination of the optimal coasting times τ^*, given the optimal maneuver sequence ω^*.

In other words, $\omega^* = \arg\min_{\omega \in L(\mathrm{MA})} \tau^*(\omega)$; for fixed ω, the optimal coasting times $\tau^*(\omega)$ can be computed solving a kinematics inversion problem, a well-studied problem for the

solution of which efficient numerical and symbolic tools are available [15], [20]. Since ω^* is a finite string, the search process will stop in a finite number of steps.

The computational efficiency of the algorithm can be vastly improved by branch-and-bound and pruning techniques. Moreover, in some cases, it is possible to partition $\mathcal{G}$ in such a way that an explicit solution for the optimal maneuver sequence is available [21].

VII. Motion Planning for a Small Helicopter

In this section, we illustrate the application of the proposed motion-planning methodology to a realistic model of an X-Cell 60 SE (manufactured by Miniature Aircraft USA, Orlando, FL) small-size helicopter. The helicopter under consideration has been instrumented at the Massachusetts Institute of Technology [22]. The helicopter is equipped with an on-board CPU and a full avionics suite, including solid-state angular rate sensors and accelerometers, GPS unit, compass, and air data system. The helicopter dynamics have been modeled using a combination of first-principle modeling and system identification, as described in [22]. The result is a nonlinear simulation that is accurate up to a forward velocity of about 20 m/s. The simulation includes several environmental disturbances and sources of errors, including wind, realistic sensor models including noise, biases, and latency, together with the actual navigation filter and control laws used in the helicopter. The simulation has been successfully validated against flight test data, even during challenging aerobatic maneuvers [22].

The state of the helicopter is described by

$$x = (R, p, \omega, v, \alpha, \beta, \omega_r)$$

where $R \in SO(3)$ is a rotation matrix representing the attitude of the helicopter, $p \in \mathbb{R}^3$ is the position of the center of mass, and $\omega \in \mathbb{R}^3$ and $v \in \mathbb{R}^3$ are, respectively, the angular and linear velocities in body axes. The scalars α and β represent the so-called rotor flapping angles in the longitudinal and lateral directions, respectively; in other words, under the assumption that the rotor blades move in a plane, α measures the component of the unit normal to the rotor disc along the longitudinal body axis, and β measures the component along the transversal body axis. Finally, the scalar ω_r represents the angular velocity of the main rotor shaft.

On board the helicopter, a rotor speed governor automatically regulates the engine throttle to maintain the rotor speed ω_r at a nominal value of 1600 rpm. The available control inputs are described by

$$u = (\delta_a, \delta_e, \delta_c, \delta_r)$$

where δ_a and δ_e are the so-called cyclic commands in the lateral and longitudinal directions, respectively; the effect of the cyclic commands is to modulate the pitch of the rotor blades during a revolution, with the effect of changing the flapping angles, and ultimately affecting mainly the moments acting on the helicopter in the roll and pitch directions. The collective command δ_c determines the average pitch of the main rotor blades, and hence, the total thrust produced by the rotor. Similarly, the "rudder" command δ_r determines the pitch of the tail rotor blades, and hence, the yawing moment generated by the tail rotor. The magnitude of each component of the control input is bounded as $\delta_a, \delta_e \in [-0.096, 0.096]$, $\delta_c \in [0.05, 0.2]$, $\delta_r \in [-0.05, 0.25]$.

The dynamics of the helicopter are described by equations of the following form:

$$\begin{aligned}
\dot{p} &= Rv, \\
\dot{R} &= R\hat{\omega} \\
\dot{v} &= -\omega \times v + R^T \bar{g} + f_v(\omega, v, \alpha, \beta, \omega_r, \delta_c, \delta_r) \\
\dot{\omega} &= -J^{-1}(\omega \times J\omega) + f_\omega(\omega, v, \alpha, \beta, \omega_r, \delta_c, \delta_r) \\
\dot{\alpha} &= f_\alpha(\omega, v, \alpha, \beta, \omega_r, \delta_a, \delta_e) \\
\dot{\beta} &= f_\beta(\omega, v, \alpha, \beta, \omega_r, \delta_a, \delta_e) \\
\dot{\omega}_r &= f_r(\omega, v, \omega_r, \delta_c, \delta_r)
\end{aligned}$$

where $\bar{g}$ is the gravity acceleration vector, J is the inertia tensor of the helicopter about its center of mass, and $\hat{\omega}$ is the unique 3×3 skew-symmetric matrix such that $\hat{\omega}u = \omega \times u$ for all $u \in \mathbb{R}^3$.

A detailed discussion of the dynamics of the helicopter is out of the scope of this paper, and can be found in the referenced sources. It will suffice, for our purposes, to discuss its invariance properties. The external forces and moments acting on the helicopter are gravity, and aerodynamic forces on the fuselage and the rotors. Assuming that the atmosphere is an isotropic and homogeneous medium, i.e., that there is no wind, and that the air density is constant with altitude, the aerodynamic forces, expressed in body axes, do not depend on the attitude or on the position of the helicopter. In other words, aerodynamic forces are invariant with respect to actions of the group of rigid body motions in space $SE(3)$. On the other hand, gravity acts along the local vertical, thus breaking this symmetry. Assuming a constant gravitational field, the gravitational forces are invariant with respect to translation and rotations about a vertical axis, i.e., to actions of the group $\mathcal{G} \cong SE(2) \times \mathbb{R} \cong \mathbb{R}^3 \times S^1$. We can identify $\mathcal{G}$ with the space of 4×4 matrices of the form

$$g(p, \psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 & p_x \\ \sin\psi & \cos\psi & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R_z(\psi) & p \\ 0 & 1 \end{bmatrix}$$

where $R_z(\psi) \in SO(3)$ represents a rotation of an angle ψ about the z axis, and $p \in \mathbb{R}^3$ is a translation. The action of $\mathcal{G}$ on the state x is defined as

$$\begin{aligned}
&\Psi\left(g(\Delta p, \Delta\psi), (R, p, \omega, v, \alpha, \beta, \omega_r)\right) \\
&\qquad = \left(R_z(\Delta\psi)R, R_z(\Delta\psi)p + \Delta p, \omega, v, \alpha, \beta, \omega_r\right).
\end{aligned}$$

Steady-state trajectories (trim primitives) correspond to arcs of helices with a vertical axis (including degenerate ones, such as straight lines and horizontal circles), flown at constant speed, constant pitch and roll angles, and constant control settings.

A. Design of a Library of Motion Primitives

The first step in the application of our methodology to motion planning for the helicopter requires the design of a library of motion primitives. This process can be carried out in several ways, for example, through model-based optimal control design, through the analysis of human-piloted flight data, or through the use of simple on-board controllers. The latter approach is taken here.

For the sake of clarity, we will consider a very small library of motion primitives, containing only four trim primitives, and seven maneuvers; in practical application, the choice of the number of motion primitives to include in the library is a matter of tradeoff between achievable performance and planning completeness and computational complexity; a typical library can contain hundreds of primitives. The planner in [13] used 625 primitives, while maintaining real-time computation capabilities.

1) Invariant Tracking Controller: An invariant tracking controller for small helicopters was introduced in [23]; this controller was implemented in the simulation environment and used both to generate motion primitives in the offline design phase, and to provide stable tracking of the nominal motion plan during actual flight, in the presence of uncertainties in the system's dynamics and disturbances in the environment. (An invariant tracking controller has the property that open-loop symmetries are preserved under feedback [24]; this is a key requirement for a tracking controller to be compatible with the approach presented in the paper.)

The controller provides, in principle, bounded tracking for the simple model of helicopter dynamics considered in [23]; however, bounds on the tracking error are not available for the realistic simulation model used in this example. In particular, it is not known *a priori* what is the class of reference trajectories that can be tracked to within a certain error bound. Hence, while it is possible, in principle, to steer the helicopter using the tracking controller alone, no guarantees would be available on the safety and on the quality of the resulting closed-loop trajectories. In particular, it would not be possible, in this case, to predict the configuration of the helicopter during its motion, e.g., in order to ensure collision avoidance, without resorting to simulation. Finally, note that the controller in [23] is not designed to minimize a physically meaningful cost.

On the other hand, embedding the helicopter's *closed-loop behavior* in the formal language discussed in Section IV makes it possible to generate trajectories that can be safely tracked, within error bounds computable *a priori*, and provides a measure of optimality with respect to a meaningful cost.

2) Trim Primitives: It is convenient to start by choosing a number of trim primitives of interest. Given the structure of the closed-loop control available on the helicopter, the generation of trim trajectories is trivial, and does not require any detailed knowledge of the actual equations of motion of the helicopter. (This statement is typically true in the vast majority of applications; most control-design methods address the problem of regulating a system around some steady-state condition.) The tracking controller itself is able to find the appropriate values of the state variables and control inputs required for trimming the helicopter at a desired altitude, velocity, sideslip, and turn rate. During the design process, only trim primitives that satisfy the flight envelope constraints are maintained and included in the library.

TABLE I
TRIM PRIMITIVE LIBRARY (PARTIAL)

ID	v [m/s]	$\dot{\psi}$ [°/s]	ϕ [°]	θ [°]	δ_r	δ_c
α	(0,0,0)	0	4.87	-0.06	0.075	0.089
β	(15,0,0)	0	4.18	-8.65	0.064	0.082
γ	(14.9, -1.43, 0)	-30	-36.4	-5.10	0.066	0.094
δ	(14.95, 0.83, 0)	30	43.4	-6.71	0.08	0.093

For the purpose of this example, four different trim primitives are chosen, namely: hover; forward level flight at 15 m/s; steady turn to the left, by 30°/s at 15 m/s flight speed; and a similar steady turn to the right. Simulation experiments provided a library of trim trajectory data, part of which is shown in Table I. In the table, we report the entries of the Lie algebra elements

$$\xi(v,\dot{\psi}) = \begin{bmatrix} 0 & -\dot{\psi} & 0 & v_x \\ \dot{\psi} & 0 & 0 & v_y \\ 0 & 0 & 0 & v_z \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

describing trim primitives, as well as the roll and pitch angles (resp., ϕ and θ), and rudder and collective inputs needed to maintain it.

3) Maneuvers: Similar considerations can be made for maneuvers; however, more design choices are available for these motion primitives. In this example, we will design a number of simple maneuvers, which entail the direct transition between two different trim primitives; moreover, we will give an example of an aerobatic maneuver.

Simple transitions between different trim primitives can be generated by commanding a transition, over a time T, in the velocity of the reference trajectory. The closed-loop behavior of the helicopter will provide a feasible trajectory achieving the desired velocity change. The choice of the time T determines the "aggressiveness" of the maneuver, and is tuned to achieve a fast response, without violating flight-envelope constraints.

Strictly speaking, since the tracking controller only provides asymptotic tracking, this procedure does not provide a valid maneuver, since maneuvers are defined as finite-time transitions between trim primitives. However, the maneuver can be truncated when the system settles at the desired trim primitives within a certain tolerance. Two examples of such maneuvers, i.e., the transition from hover to forward flight at 15 m/s, and from forward flight to turning flight, are provided in Figs. 2 and 3. Partial state and control trajectories during the execution of a motion plan are reported in Figs. 4 and 5. The invariance of the system with respect to time and group actions are evident in these figures; minor deviations and the high-frequency "noise" are due to environmental disturbances (e.g., irregularities in the engine power output) and numerical issues.

The MA language presented in this paper also allows us to include aerobatic maneuvers in the library of motion primitives. In this example, we will consider a maneuver called "ag turn," similar to the aerobatic figure known as "hammerhead," which allows the helicopter to quickly reverse the direction of flight,

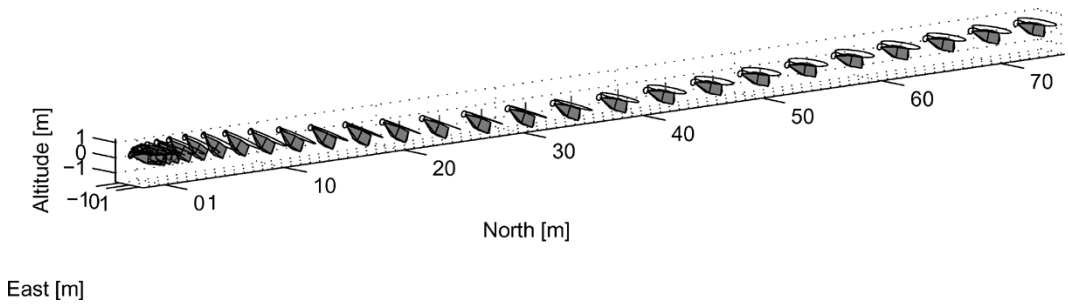

Fig. 2. Maneuver example. Transition from hover to forward flight at 15 m/s (top). The helicopter is shown at intervals of 0.25 s.

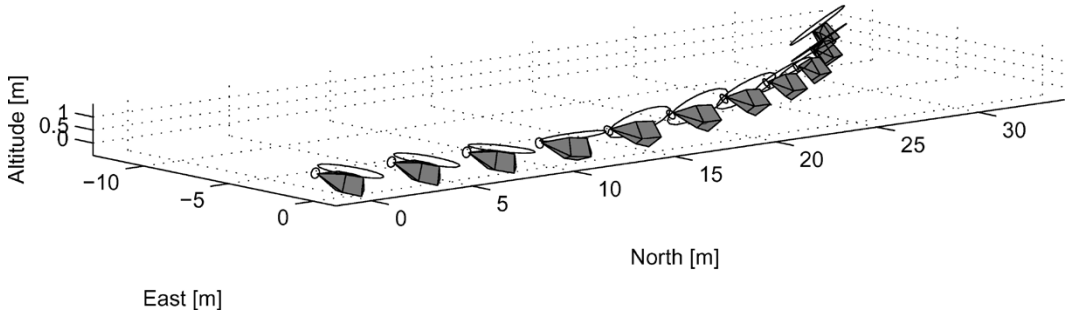

Fig. 3. Maneuver example. Transition from forward flight at 15 m/s to steady turning flight to the left, at a turning rate of 30°/s and a velocity of 15 m/s.

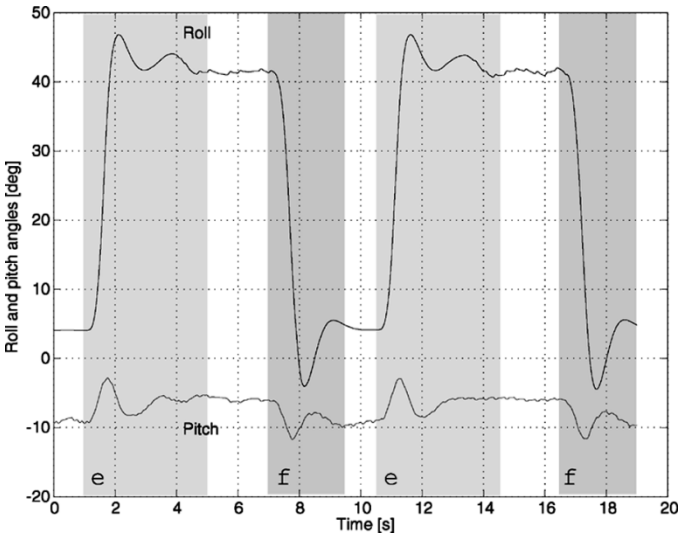

Fig. 4. Roll and pitch angle history during the execution of the motion plan $p_0 = (\texttt{efef}, [1, 2, 1, 2, 0])$. The shaded bands identify maneuver execution times (the maneuver id is reported at the bottom left of each band).

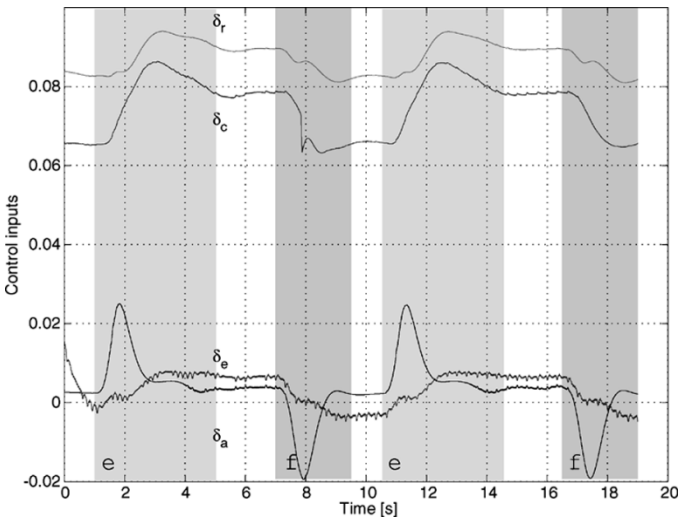

Fig. 5. Control input history during the execution of the motion plan $p_0 = (\texttt{efef}, [1, 2, 1, 2, 0])$.

in a very tight space, and with minor changes in velocity and altitude before and after the maneuver. (The name of this maneuver stands for "agricultural turn;" such a maneuver is used by crop-dusting pilots to optimize the flight path.)

As illustrated in Fig. 6, this maneuver is initiated with a pull-up, trading airspeed for altitude. At the apex of the trajectory, when the velocity is reduced to almost zero, a yaw rotation is performed, reversing the heading of the helicopter. A dive then ensues, ending at approximately the original altitude and velocity, but with opposite heading; this is achieved in about 5 s, starting from forward flight conditions at 15 m/s (i.e., from the trim primitive β). A 2-s altitude recovery phase follows, during which the helicopter climbs back to the altitude it had before the turn. (According to the simulation, the vehicle loses 3 m of altitude in the course of the ag turn maneuver.)

According to the above discussion, maneuvers are designed to construct a strongly connected graph, as depicted in Fig. 7. For each of these maneuvers, information on the time duration and the group displacement are presented in Table II. (In addition, the full state and control trajectories defining a prototype of such a primitive are stored in the motion primitive library.)

B. Motion Planning

1) Controllability Check: Before proceeding with examples of motion-planning problems, we need to ensure that the collection of motion primitives defined in the preceding section is indeed sufficient for controllability. (All primitives are designed

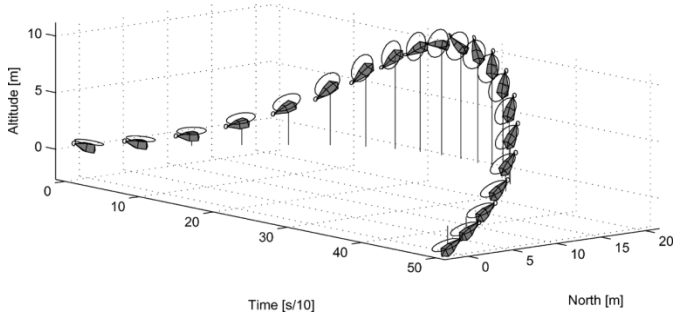

Fig. 6. Profile of the "ag turn" maneuver referenced in the text, and included in the MA in the example. For the sake of clarity, one of the axes in the figure represents time; the entirety of the maneuver is executed within 1 m in the east–west direction. The helicopter is shown at 0.25-s intervals along its flight path.

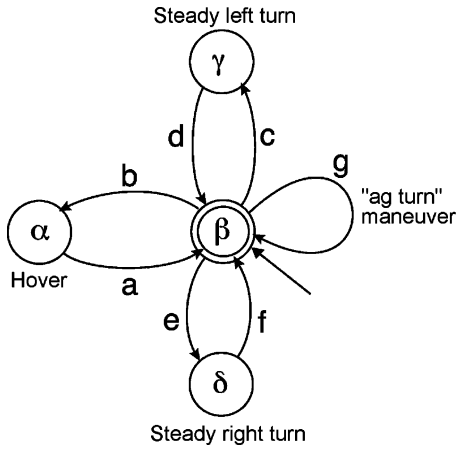

Fig. 7. MA discussed in the example.

TABLE II
MANEUVER LIBRARY (PARTIAL)

ID	Pred	Succ	Duration [s]	Δp	$\Delta\psi$[°]
a	α	β	7.5	(67.5, 0, 0)	0
b	β	α	5	(22.5, 0 0)	0
c	β	γ	4.5	(31.5, -41.7.0, 0)	-120.0
d	γ	β	2	(28.9, -6.6, 0)	-15.0
e	β	δ	4	(34.2, 34.9, 0)	105.0
f	δ	β	2.5	(36.1, 8.6, 0)	15.0
g	β	β	7.1	(-43.5, 0, 0)	180

with no net altitude change; hence, we will restrict ourselves to motion-planning problems on the plane, even though the dynamics of the helicopter are fully 3-D, and altitude changes occur during the execution of maneuvers.) First of all, the digraph in Fig. 7 is strongly connected, and the group of rigid motions on the plane is connected. Consider the motion plan $p_0 = (\texttt{efef}, [1,2,1,2,0])$. From the data reported in Tables I and II, it can be verified that

$$g_{p_0} = \exp(\xi_\beta) g_{\mathtt{e}} \exp(2\xi_\delta) g_{\mathtt{f}} \exp(\xi_\beta) g_{\mathtt{e}} \exp(2\xi_\delta) g_{\mathtt{f}} = e$$

i.e., that p_0 is a fixed-point motion plan. Furthermore, the vectors $\eta_{0,i}$, defined in (11), span the subalgebra $\mathfrak{se}(2) \subset \mathfrak{se}(3)$. Hence, from *Theorem 5.1*, we can conclude that our collection of primitives is indeed sufficient to steer the system to any configuration at the same altitude as the initial conditions.

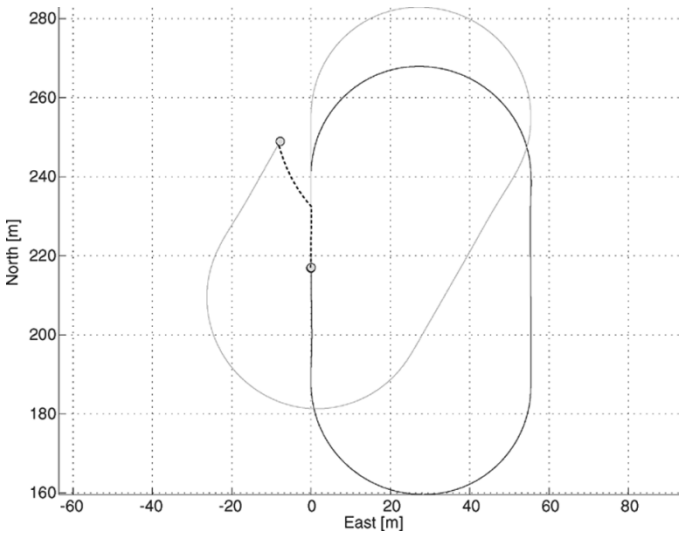

Fig. 8. Trajectory flown during the execution of the motion plan $p = (\texttt{efefefef}, [1,2,1,2,1+1,2+1,1,2,0])$. The fixed-point motion plan p_0 mentioned in the text is contained in p. The dashed line shows the composition of unit flows along $\eta_{0,1}$ and $\eta_{0,2}$.

In order to gain some intuition on how to control the configuration of the helicopter using perturbations on the fixed-point motion plan p_0, in Fig. 8, we report the northeast trajectory corresponding to p_0 (dark curve), followed by a motion plan equal to $(\omega_0, \tau_0 + (1,1,0,0,0))$. The final group displacement can be computed as

$$g_f = \exp \eta_{0,1} \exp \eta_{0,2} = \begin{bmatrix} \frac{\sqrt{3}}{2} & -\frac{1}{2} & 30.9 \\ \frac{1}{2} & \frac{\sqrt{3}}{2} & -7.5 \\ 0 & 0 & 1 \end{bmatrix}.$$

The flow along these vector fields is also shown in Fig. 8.

2) Feasible Motion Plan Computation: The same fixed-point motion plan we used to check controllability can be used to compute a feasible motion plan between two arbitrary configurations of the form $\Psi(g, x_q(0))$, with $g \in SE(2)$, and $q \in Q \setminus \{\square\}$. This is due to the fact that the composition of positive flows along the left-invariant vector fields identified by $\eta_{0,i}$ spans the whole group $SE(2)$.

For example, consider the following steering problem. Steer the helicopter from the initial state $x_0 = x_\beta(0)$ (northward flight at the origin) to the target state $x_f = \Psi(g_f, x_\alpha(0))$, with

$$g_f = \begin{bmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & 0 \\ -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & -100 \\ 0 & 0 & 1 \end{bmatrix}$$

i.e., forward flight with a $-45°$ (northwest) heading, at position (0,−100).

A feasible motion plan can be found if there exist nonnegative coasting times τ_i that solve the following equation:

$$g_f = g_{\mathtt{e}} g_{\mathtt{f}} g_{\mathtt{e}} g_{\mathtt{f}} \prod_{i=1}^{5} \exp(\eta_i \tau_i)$$

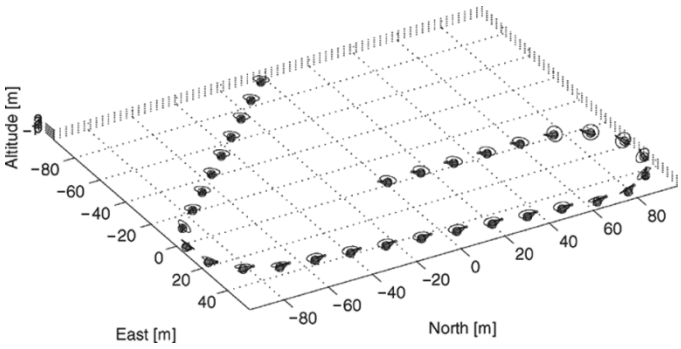

Fig. 9. Feasible motion plan for the example problem, constructed from the fixed-point motion plan used to prove controllability.

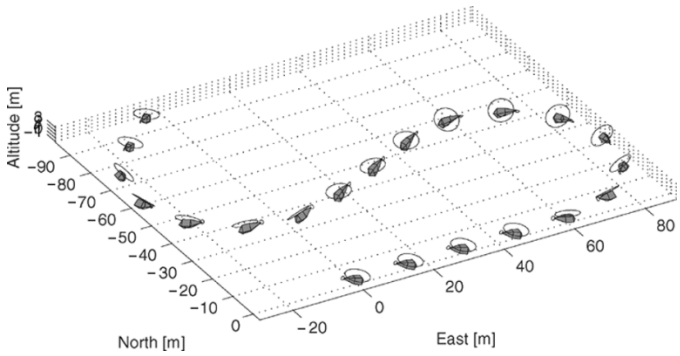

Fig. 10. Improved motion plan for the example problem.

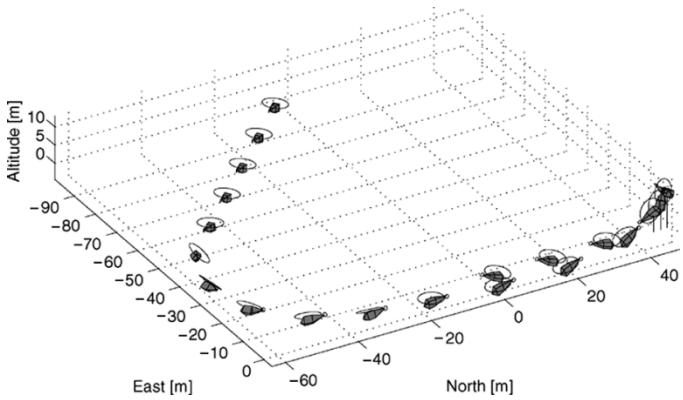

Fig. 11. Optimal motion plan for the example problem.

that is

$$\prod_{i=1}^{5} \exp(\eta_i \tau_i) = \begin{bmatrix} \cos\left(\frac{5}{6\pi}\right) & -\sin\left(\frac{5}{6\pi}\right) & 103.2 \\ \sin\left(\frac{5}{6\pi}\right) & \cos\left(\frac{5}{6\pi}\right) & 117.5 \\ 0 & 0 & 1 \end{bmatrix}.$$

The above underdetermined equation, involving linear and trigonometric functions of the coasting times τ_i, can be written as a polynomial equation through the identities $\sin\theta = 2t/(1+t^2)$, $\cos\theta = (1-t^2)/(1+t^2)$, with $t = \tan(\theta/2)$. A solution is found as $p_{\text{feas}} = (\texttt{efef}, [4.1, 2.3, 6.6, 0.23, 6.3])$; the corresponding trajectory is shown in Fig. 9.

3) Optimal Motion-Plan Computation: The total time needed to execute the motion plan p_{feas} is equal to 32.5 s. It is now desired to minimize the total time to steer the helicopter between the two specified configurations.

In order to do this, we need to consider all maneuver sequences $\omega \in L(\text{MA})$, where the initial state of the automaton MA is $q_0 = \beta$, and the set of accepting states is $\mathcal{F} = \{\beta\}$, such that $|\omega| < |p_{\text{feas}}|$. For each of these maneuver sequences, we need to solve the optimization problem (13).

While the number of such maneuver sequences is, in principle, large, bounding and constraint satisfaction arguments result in a substantial pruning of the decision tree describing all feasible maneuver sequences. For example, one can immediately conclude that optimal maneuver sequences will not include any substring of the form $(\texttt{ab})$; the trim primitive β executed for 6 s results in the same group displacement, and has the same pre- and postconditions, with a lower cost. Maneuver sequences must include at least one substring of the form $(\texttt{cd})$ or $(\texttt{ef})$ in order to set the final heading angle to the desired value.

The maneuver sequences `cd` and `ef` do not solve the steering problem, as program (13) is infeasible in these cases. Using the maneuver sequence `cdef`, (13) yields the motion plan $p_3 = (\texttt{cdef}, [3.36, 2.37, 0.56, 1.37, 0])$, with a cost of 20.68 s. The cost of p_3 can now be used as an upper bound on the cost of candidate optimal maneuver sequences, instead of $|p_{\text{feas}}|$. The corresponding trajectory is shown in Fig. 10. The optimal maneuver-based motion plan is eventually found to be $p_{\text{opt}} = (\texttt{gef}, [1.72, 0.55, 0.5, 2.96])$, with a cost $|p_{\text{opt}}| = 19.31$ s. As evident from the optimal maneuver sequence, this motion plan includes the "ag turn" maneuver; Fig. 11 shows the trajectory flown by the helicopter.

We remind the reader that "optimality" here must be understood under the constraints of the available motion primitives, i.e., this motion plan is not optimal when the continuous dynamics of the helicopter are considered. On the other hand, the computation of this motion plan, making optimal use of precomputed primitives, took about half a second on a 867 MHz PowerPC G4, running Mac OS X. We remark that a hard bound can be given for the computation time for obtaining a *feasible* motion plan; this is particularly important for real-time applications. In our case, a feasible motion plan can be obtained in about three-hundredths of a second.

Finally, to our knowledge, our motion-planning methodology is unique in its ability to include "aggressive" or acrobatic maneuvers in a motion plan, as shown in the example. (The solution of the full optimal control could also include such maneuvers, but at a prohibitive computational cost.)

VIII. Conclusions and Future Work

In this paper, we introduced a formal language for motion description, and computationally efficient algorithms for computing solutions to a certain class of motion-planning problems. The proposed approach is applicable to a large class of nonlinear systems with symmetries, including mobile robots and autonomous vehicles.

The key advantages of the proposed approach are two-fold: all motion plans generated according to the rules of the language are implicitly feasible with respect to the dynamics of

the system; and safe with respect to the operational-envelope constraint. Furthermore, the map relating the initial and final configurations of the system has the same structure of a forward kinematic map, even for systems with complex dynamics. As a consequence, tools developed for motion planning of kinematic systems can be brought to bear onto problems involving dynamical systems, without resorting to approximations or simplifications. Exploiting symmetries, one can extend the "local" knowledge of a set of feasible trajectories, and possibly of locally stabilizing control laws, to achieve global goals. As shown in the example, the method can be successfully applied to very complicated, realistic models of challenging dynamical systems, such as small aerobatic helicopters.

Future work will include a more detailed treatment of the robustness issues associated with the application of the proposed framework to systems characterized by uncertainty in the dynamics and environmental disturbances. As long as the uncertainties and disturbances can be modeled as being invariant under the action of the group $\mathcal{G}$, most of the properties of Maneuver Automata and of the corresponding languages are indeed preserved.

Another direction of future work includes the analysis of the suboptimality gap induced by reducing feasible trajectories to the sequential combination of motion primitives. This will lead to an accurate tradeoff between the complexity of the MA language, and consequently, of the algorithms defined on it, and the achievable performance. For example, it is of interest to quantify the maximum penalty which must be accepted, given a certain set of motion primitives. Conversely, it is of interest to determine the optimal (e.g., minimal) choice of motion primitives which guarantee a certain optimality gap.

ACKNOWLEDGMENT

Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

REFERENCES

[1] A. E. Bryson and Y. C. Ho, *Applied Optimal Control*. New York: Hemisphere, 1975.
[2] S. Sastry, *Nonlinear Systems: Analysis, Stability, and Control*. New York: Springer-Verlag, 1999, vol. 10.
[3] V. Gavrilets, E. Frazzoli, B. Mettler, M. Piedmonte, and E. Feron, "Aggressive maneuvering of small autonomous helicopters: A human-centered approach," *Int. J. Robot. Res.*, vol. 20, no. 10, pp. 795–807, 2001.
[4] D. F. Delchamps, "Stabilizing a linear system with quantized state feedback," *IEEE Trans. Autom. Control*, vol. 35, no. 8, pp. 916–926, Aug. 1990.
[5] A. Marigo and A. Bicchi, "Steering driftless nonholonomic systems by control quanta," in *Proc. IEEE Conf. Decision Control*, vol. 4, 1998, pp. 4164–4169.
[6] R. W. Brockett and D. Liberzon, "Quantized feedback stabilization of linear systems," *IEEE Trans. Autom. Control*, vol. 45, no. 6, pp. 1279–1289, Jun. 2000.
[7] N. Elia and S. K. Mitter, "Stabilization of linear systems with limited information," *IEEE Trans. Autom. Control*, vol. 46, no. 9, pp. 1384–1400, Sep. 2001.
[8] R. W. Brockett, "Formal languages for motion description and map making," in *Robotics*, R. W. Brockett, Ed. Providence, RI: Amer. Math. Soc., 1990, vol. 41, pp. 181–193.
[9] V. Manikonda, P. S. Krishnaprasad, and J. Hendler, "Languages, behaviors, hybrid architectures and motion control," in *Mathematical Control Theory*, J. Bailleul and J. C. Willems, Eds. New York: Springer, 1998, pp. 199–226.
[10] M. Egerstedt and R. W. Brockett, "Feedback can reduce the specification complexity of motor programs," *IEEE Trans. Autom. Control*, vol. 48, no. 2, pp. 213–223, Feb. 2003.
[11] D. Hristu-Varsakelis, M. Egerstedt, and P. S. Krishnaprasad, "On the structural complexity of the motion description language MDLe," in *Proc. IEEE Conf. Decision Control*, vol. 4, Maui, HI, 2003, pp. 3360–3365.
[12] L. Kovar, M. Gleicher, and F. Pighin, "Motion graphs," *ACM Trans. Graphics (Special Issue: Proc. ACM SIGGRAPH 2002)*, vol. 21, no. 3, pp. 473–482, Jul. 2002.
[13] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *AIAA J. Guid., Control, Dynam.*, vol. 25, no. 1, pp. 116–129, 2002.
[14] A. Bicchi, A. Marigo, and B. Piccoli, "On the reachability of quantized control systems," *IEEE Trans. Autom. Control*, vol. 47, no. 4, pp. 546–563, Apr. 2002.
[15] R. Murray, Z. Li, and S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL: CRC, 1994.
[16] F. Bullo and K. M. Lynch, "Kinematic controllability for decoupled trajectory planning in underactuated mechanical systems," *IEEE Trans. Robot. Autom.*, vol. 17, no. 4, pp. 402–412, Aug. 2001.
[17] H. Sussmann and V. Jurdjevic, "Controllability of nonlinear systems," *J. Differential Equations*, vol. 12, pp. 95–116, Jul. 1972.
[18] B. Jakubczyk and E. D. Sontag, "Controllability of nonlinear discrete time systems: A Lie-algebraic approach," *SIAM J. Control Optim.*, vol. 28, pp. 1–33, 1990.
[19] E. Frazzoli, *Robust Hybrid Control for Autonomous Vehicle Motion Planning*. Cambridge, MA: Massachusetts Inst. Technol., Jun. 2001.
[20] D. Manocha and J. F. Canny, "Efficient inverse kinematics for general 6R manipulators," *IEEE Trans. Robot. Autom.*, vol. 10, no. 5, pp. 648–657, Oct. 1994.
[21] E. Frazzoli, "Explicit solutions for optimal maneuver-based motion planning," in *Proc. IEEE Conf. Decision Control*, vol. 4, Maui, HI, 2003, pp. 3372–3377.
[22] V. Gavrilets, B. Mettler, and E. Feron, "Nonlinear model for a small-size acrobatic helicopter," in *Proc. AIAA Guid., Navig., Control Conf.*, Montreal, QC, Canada, 2001, Paper AIAA-2001-4333.
[23] E. Frazzoli, M. Dahleh, and E. Feron, "Trajectory tracking control design for autonomous helicopters using a backstepping algorithm," in *Proc. Amer. Control Conf.*, vol. 6, Chicago, IL, 2000, pp. 4102–4107.
[24] P. Rouchon and J. Rudolph, "Invariant tracking and stabilization: Problem formulation and examples," in *Stability and Stabilization of Nonlinear Systems*, D. Aeyels, F. Lamnabhi-Lagarrigue, and A. van der Schaft, Eds. London, U.K.: Springer-Verlag, 1999, vol. 246, pp. 261–273.

Emilio Frazzoli (S'99–A'01) received the Laurea degree in aerospace engineering from the University of Rome "La Sapienza," Rome, Italy, in 1994, and the Ph.D. degree in navigation and control systems from the Massachusetts Institute of Technology, Cambridge, in 2001.

Between 1994 and 1997 he was an officer in the Italian Navy, and a spacecraft dynamics specialist with the European Space Agency, Darmstadt, Germany, and Telespazio, Rome, Italy. From 2001 to 2004, he was an Assistant Professor of Aerospace Engineering with the University of Illinois at Urbana-Champaign. Since 2004, he has been an Assistant Professor of Mechanical and Aerospace Engineering with the University of California, Los Angeles. His current research interests include algorithmic, computational, and geometric approaches to the design and development of decision and control architectures for complex networked and autonomous systems, in aerospace and other domains. Application areas include distributed cooperative control of multiple vehicle systems, guidance and control of agile vehicles, high-confidence software engineering for high-performance dynamical systems, and verification of hybrid systems.

Munther A. Dahleh (F'00) was born in 1962. He received the B.S. degree from Texas A & M University, College Station, TX, in 1983, and the Ph.D. degree from Rice University, Houston, TX, in 1987, all in electrical engineering.

Since then, he has been with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, where he is now a full Professor. He was a Visiting Professor with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA, during the spring of 1993. He has held consulting positions with several companies in the U.S. and abroad. His research interests include problems at the interface of robust control, filtering, information theory, and computation, which include control problems with communication constraints and distributed mobile agents with local decision capabilities. He is also interested in model-reduction problems for discrete-alphabet hidden Markov models, universal learning approaches for systems with both continuous and discrete alphabets, and in providing an anatomically consistent model of the motor control system.

Dr. Dahleh was the recipient of the Ralph Budd award in 1987 for the best thesis at Rice University, the George Axelby outstanding paper award (paper coauthored with J. B. Pearson in 1987), an NSF Presidential Young Investigator Award (1991), the Finmeccanica Career Development Chair (1992), the Donald P. Eckman award from the American Control Council in 1993, the Graduate Students Council teaching award in 1995, and the George Axelby outstanding paper award (paper coauthored with Bamieh and Paganini in 2004).

Eric Feron is an Associate Professor of Aeronautics and Astronautics with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge. His research interests include optimization and control theory and their applications to aerial robotics, air transportation and software engineering. His latest book is a translation of Etienne Bézout's *General Theory of Algebraic Equations*, which sets the foundation for modern linear algebraic techniques in polynomial optimization theory. His latest research achievements include autonomous aerobatic flight of small helicopters and a natural language interface to unmanned aerial vehicles (built with Teragram Corporation and Boeing).