ON COMPUTABILITY OF FINE MOTION PLANS*

John Canny

Computer Science Division
University of California, Berkeley
Berkeley, California 94720

1 Abstract

We show that fine motion plans in the LMT framework developed by Lozano-Pérez, Mason and Taylor are computable, and we give an algorithm for computing them by reducing fine motion planning to an algebraic decision problem. *Fine Motion Planning* involves planning a successful motion of a robot at the fine scale of assembly operations, where control and sensor uncertainty are significant. We show that as long as the envelope of trajectories generated by the control system can be described algebraically, there is an effective procedure for deciding if a successful n-step plan exists. Our method makes use of *recognizable sets* as subgoals for multi-step planning. These sets are finitely parametrizable, and we show that they are the only sets that need be considered as subgoals. Unfortunately, if the full generality of the LMT framework is used, finding a fine motion plan can take time double exponential in the number of plan steps.

2 Introduction

This paper deals with fine motion planning as formalized by Lozano-Perez, Mason and Taylor [12]. In their original paper, these authors presented a new framework for fine motion planning that explicitly took into account both sensor and control uncertainty. An integral part of their methodology was a force control scheme, either a generalized damper or generalized spring. The [12] paper gave a very abstract description of a fine motion planning method, and it was not clear that such plans were computable. Subsequent work has studied restrictions to the LMT model that lead to computable plans [7,11], but a significant amount of power is lost with these restrictions. In particular, the plan executor cannot make use of time or sensor history. In this paper, we show that LMT plans are computable in full generality, as long as the trajectory envelope can be described algebraically. We do not know as yet whether the addition of continuous sensor history as proposed by Mason [14] (instead of the finite sensor history used in [12]) still gives a computable planning problem.

In [3] it was shown that LMT plans may be very long. Specifically, it was shown that even in a three dimensional polyhedral environment, some plans require a number of plan steps that grows exponentially with the environment complexity. It was also shown that finding a guaranteed plan is hard for non-deterministic exponential time, which is strong evidence that finding plans can take double exponential time in the worst case. In this paper, the number of plan steps n is fixed, and is input to the algorithm. We believe that the lower bounds just mentioned should not be of concern, because it is only realistic to consider fine motion plans with a small number of steps.

What we give here is a computability result, but the time bounds are poor. Previously, Donald had shown that LMT plans are computable in the plane without history or time [6]. Finding an n-step plan can take time double exponential in n, i.e. the running time is of the form $O(2^{2^n})$. This bound follows because we can reduce the fine motion planning problem to a decision problem in the theory of the real numbers, with a number of real quantified variables proportional to n. Deciding formulae in the theory of the reals takes double exponential time in the number of quantifiers. On the positive side, the algebraic decision algorithms are parallelizable, and the parallel time is single exponential in the number of quantifiers. Of course the number of processors is double exponential and is completely prohibitive. But worst case bounds do not imply long running times in typical cases, and what we describe here can be used as a starting point for more efficient average-case planners.

The reduction from fine motion planning produces an algebraic formula with alternating existential and universal real quantifiers. The existential quantifiers represent the actions taken by the plan executor, like choosing the direction to move or the time to stop moving, and the universals represent the actions of nature, namely the sensor readings. This game-theoretic interpretation of fine motion planning was explored in [16]. It is the alternation of these quantifiers that leads to the double exponential time bound.

We begin our exposition with two introductory sections, one on the algebraic tools needed for the computability result, and the other on the LMT fine motion framework. Section 5 contains our main result, and introduces recognizable sets as the basic subgoals in multi-step plans.

3 Real Algebraic Algorithms

The basic building blocks that we use to represent geometric objects are formally called semi-algebraic (SA) sets. A very broad range of geometric objects can be represented exactly as SA sets. A few common objects however, those whose boundaries are not algebraic, must be approximated.

SA sets are defined by predicates which are logical combinations of polynomial inequalities. They can be defined formally in a real space $\Re^m$ whose coordinates are $x_1, \ldots, x_m$ via:

Definition A *semi-algebraic* (SA) set is the set of points in the real space $\Re^m$ that satisfy a predicate of the form $B(Q_1(x_1, \ldots, x_m), \ldots, Q_k(x_1, \ldots, x_m))$ where $B : \{T, F\}^k \rightarrow \{T, F\}$ is a k-argument binary function, and each $Q_i(x_1, \ldots, x_m)$ is polynomial inequality. That is, each

*Sponsored by the Defense Advanced Research Projects Agency (DoD), monitored by Space and Naval Warfare Systems Command under Contract N00039-88-C-0292.

Q_i is of the form $P_i(x_1, \ldots, x_m) \oplus 0$, where P_i is a polynomial, and $\oplus$ can be any of the comparators $=, \neq, >, <, \geq$ or $\leq$. We call the defining predicate an *SA predicate.*

Polyhedra are a special case of SA sets that have linear defining polynomials P_i. Constructive solid geometry (CSG) models are also SA sets. Objects that have algebraic parametric surface patch boundaries are SA sets, for example, objects with B-spline boundaries.

3.1 Quantifier Elimination

The most important property of semi-algebraic sets is that they are closed under projection, and that this is a computable operation. This means that if we have a universal or existentially quantified real variable in front of an SA predicate, we can eliminate this quantifier, and obtain a new SA predicate in the free variables.

Theorem 3.1 *(Tarski) If $A(x_1, \ldots, x_m)$ is an SA predicate, then so is $B(x_1, \ldots, x_{m-1}) = \exists x_m\ A(x_1, \ldots, x_m)$ and $C(x_1, \ldots, x_{m-1}) = \forall x_m\ A(x_1, \ldots, x_m)$. Both the predicates B and C are quantifier-free SA predicates.*

This theorem for single quantifiers implies that sequences of quantifiers can be eliminated, starting from the inside out. For example, by eliminating z, $\forall x\ \exists y\ \forall z\ A(x, y, z)$ becomes $\forall x\ \exists y\ A'(x, y)$ which becomes $\forall x\ A''(x)$, which evaluates to either true or false.

Algorithms for doing quantifier elimination on SA predicates have been around for some time, starting with Tarski's paper [15]. The best sequential time bounds are due to Collins [4], who showed that quantifier elimination can be done in time double-exponential in the number of variables m, that is, the running time of his algorithm is roughly 2^{2^m}.

Somewhat better bounds are possible if the quantifiers are of a particular type, e.g. all existential. If this is the case, they can be eliminated in parallel polynomial time [2]. Or even if the number of alternations of quantifier type is fixed, e.g. a string of universals followed by a string of existentials followed by universals counts as two alternations, the sequential time bound drops to single exponential [9].

3.2 Sample Points

If we have an SA predicate that defines a set A which is non-empty, we sometimes need to find a point p which actually lies in the set. Collin's method [4] produces such sample points. The author's method [2] produces "symbolic" sample points, but numerical coordinates of these points can be found by solving a single univariate polynomial. It is somewhat faster than Collin's method for multidimensional sets. Of course, if all the points in A are irrational, then an exact sample point cannot be found. However, it is possible with the above schemes to get arbitrarily close to the coordinates of a true sample point.

So it can be assumed that if one has a predicate defining a non-empty SA set, there is an effective procedure for finding a numerical sample point (or approximation) in the set. Both methods have running times that are polynomial in the size of the SA predicate, but the exponents grow with the dimension of the set. In Collins method the exponent is exponential in the dimension, while in the author's method, it is linear in dimension.

3.3 Set Notation

In the following sections, to simplify our exposition, we will often refer to a semi-algebraic set without explicitly giving its defining predicate. For a set $A \subseteq \Re^m$, there is an implicitly defined predicate $A(x_1, \ldots, x_m)$ which is true at a point iff that point is in A. For two sets, $A \cap B$ has defining predicate $A(x_1, \ldots, x_m) \wedge B(x_1, \ldots, x_m)$, and similarly for $A \cup B$ etc.

We will also write phrases like $A \subseteq B$, which is a logical statement that translates into a quantified formula in the corresponding predicates:

$$\forall x_1, \ldots, x_m\ A(x_1, \ldots, x_m) \Rightarrow B(x_1, \ldots, x_m) \qquad (1)$$

and when we write $p \in A$ where p is a point with coordinates $p_1, \ldots, p_m$, we mean that the statement $A(p_1, \ldots, p_m)$ is true.

For compactness, if p has m real coordinates, we will write $\forall p$ instead of $\forall p_1, \ldots, p_m$, etc.

Finally, we will often make use of parametrically-defined sets $A(q) \subset \Re^m$, where $A(q)$ is an *SA set* which is a function of some argument q. This means that there is an implicitly defined SA predicate $A(p, q)$ which is true iff $p \in A(q)$. So when we write $A(q) \subseteq B$, we mean $\forall p\ A(p, q) \Rightarrow B(p)$ which gives us an *SA predicate* $H(q)$ which is true of those values of q such that $A(q) \subseteq B$. We will be careful to state whether a symbol describes a predicate or a set when this is not clear from the context.

4 LMT Fine Motion Planning

Most commercial robots use position control only. Because position control errors are significant for many assembly tasks, robot applications are biased toward low-precision tasks such as welding and spray-painting. To successfully plan high-precision assemblies, uncertainty must be taken into account, and other types of control must be used which allow *compliant motion.* Compliant motion occurs when a robot is commanded to move into an obstacle, but rather than stubbornly obeying its motion command, it complies to the geometry of the obstacle.

Compliant motion ([10], [13]) is possible only with certain dynamic models. The two most common of these are the generalized spring and generalized damper models, [13]. Our methods apply to either model, but we will sketch only the generalized damper model. The generalized spring was studied in the context of LMT planning in [1]. We do not need the details of the control scheme in our planner, but the planning algorithm does require that the trajectory envelope be describable algebraically, and this description *will* depend on the control scheme. We give such a description for a typical damper control scheme and a three-dimensional environment.

LMT plans can be summarized as follows: A plan consists of n steps. At the begining of each step, the plan executor makes a sensor reading s_0^* of position and velocity, and it chooses a commanded motion θ. The commanded motion is passed to the control system, usually a generalized damper or generalized spring, which moves the robot along some trajectory consistent with the motion command. Because of control uncertainty, this trajectory is not unique, and is not known (except indirectly through sensor readings) to the plan executor. The trajectory will typically involve sliding against obstacles. As the robot moves, at each time t, the plan executor takes a sensor reading s_1^* and evaluates a *termination predicate* $P(s_0^*, \theta, t, s_1^*)$. This predicate should return a true value if and only if the robot has successfully entered a goal region (or intermediate subgoal region). If a subgoal is reached and recognized by the termination predicate, the next plan step commences, using s_1^* as the initial sensor reading.

The framework takes into account the inaccuracy in sensor readings and in the control system, so that the actual

state of the robot is not known. Instead, the plan executor has partial knowledge of state, i.e. it knows that the robot's state is in some set of states that are consistent with the sensor readings. When the termination predicate returns a true value, it means that it has determined that *all possible* actual states of the robot that are consistent with the sensor readings lie in the goal region.

We assume that we are working in the configuration space of the robot, so that the robot itself is always a point. Let $\mathcal{C}$ denote the configuration space, the set of free (obstacle avoiding) configurations $F \subset \mathcal{C}$ of the robot is assumed to be an SA set.

In the damper control scheme, during each step of the fine motion plan, the point robot is commanded to move at some velocity v^c. Because of control uncertainty however, the point actually moves with a control velocity v which lies in a ball of radius $\epsilon||v^c||$ about the commanded velocity, i.e.

$$||v - v^c|| < \epsilon||v^c|| \tag{2}$$

where $||\cdot||$ is a suitable norm. In the LMT model, the control velocity v can change instantaneously, as long as it always remains within the error ball about v^c. This description gives us a complete characterization of the robot's motion in free space, but during contact, we need to take the dynamic model into account.

The generalized damper applies forces to the robot in response to velocity errors. If F is the force on the robot, v the control velocity and v^a the actual velocity, the damper equation is

$$F = B(v^a - v) \tag{3}$$

where B is a matrix called the *damping matrix*. Assuming B has all negative eigenvalues, this gives a stable (idealized) negative feedback system, and in free space we find that $v^a = v$ as we would expect.

Motion in contact with a surface is possible if the control velocity v is directed into the surface. The actual velocity v^a will be tangent to the surface in this case, since the reaction force will cancel the component of F directed into the surface. We omit the details of derivation of v^a in this case, but refer the reader to [7]. It is enough for us to know that both the velocity v^a, and the robot's trajectory as a function of time can be specified using SA predicates.

4.1 Feasible Trajectories

Having found the actual velocities that occur in response to a control velocity, we can now try to represent trajectories of the robot over time. In the original LMT framework, the control velocity v can change instantaneously, as long as it is consistent with the commanded velocity v^c. This is a very strong assumption, and makes the set of trajectories virtually impossible to represent. However, if we assume that the control velocity changes only a number of times proportional to the number of environment surfaces, we find that in most cases we can generate the same trajectory envelope as if we allowed an infinite number of changes.

We define a trajectory T as a map from time to position times velocity state space. At each time t, $T(t)$ is a pair (p, v) which are respectively the position and velocity of the robot at time t. If $\mathcal{C}$ is the configuration space of the robot, and $T\mathcal{C}$ is its tangent bundle,

Definition A *trajectory* is a map $T : [0, t_{max}] \rightarrow T\mathcal{C}$ such that $T(t) = (p, v)$ where p and v are the position and velocity respectively of the robot at time t. The position coordinate of a trajectory is a continuous function of time, but velocity may have discontinuities.

We saw in the last section that the motion of the robot (i.e. its actual velocity) is completely determined by the control velocity. In the present paper, we assume that rather than varying continuously, the control velocity takes on m values $v_1, \ldots, v_m$, and there are $m - 1$ time switchpoints t_k, where t_k is the time at which the control velocity switches from v_k to v_{k+1}. We do not justify this here, but it is the case that for the polyhedral environment described below that the trajectory envelope with a finite number of switchpoints is the same as if the number of switchpoints is infinite. In general we feel this is a reasonable compromise between the ideal model and a computable formalization. We also feel that it is a reasonable reflection of what happens in the physical world in response to a motion command.

Definition We write $\mathcal{T}(s_0, \theta)$ for the set of *feasible trajectories* T with initial state $T(0) = s_0$ that are consistent with the motion command θ.

Each trajectory is determined by a finite number of real values t_k and v_k as described above, together with s_0. The set $\mathcal{T}(s_0, \theta)$ defines the values of these parameters that correspond to feasible trajectories, and is an SA subset of the set of all parameter values. For the damper, the motion command θ is just the actual commanded velocity v^c, and the feasible parameters are just the v_k that lie within the uncertainty ball about v^c.

While the sequences (v_k), (t_k) determine the trajectory T, they do not give a convenient representation of it. A better representation is to give the actual velocities along the segments, together with the positions at the switchpoints. To do this, we need to take into account the interaction of the robot with the environment.

In the special case of a polyhedral environment in three dimensions, we have $\mathcal{C} = \Re^3$ and $T\mathcal{C} = \Re^6$, and each actual velocity segment v_k^a lies either entirely in free space, or is a straight line along some obstacle face, or along a concave edge. We assume that we have an SA predicate $F(x)$ defining free space, and that at each point x on the boundary of free space, the normal $N(x)$ is known.

Then, given a control velocity v_k, and knowing the position of the robot p_k at time t_k, we can determine if the robot will slide or move through free space during this segment. If it slides, from the normal information, we can determine its actual velocity v_k^a, and therefore its position p_{k+1} at time t_{k+1}. In this last step we assume that each segment involves a single type of motion, that is, between t_k and t_{k+1}, the robot is either entirely in free space, or is sliding against a single surface or edge. We add this as a restriction to the class of feasible trajectories $\mathcal{T}(s_0, \theta)$. $\mathcal{T}(s_0, \theta)$ will still be an SA predicate in the values of $v_1, \ldots, v_m$ and $t_1, \ldots, t_{m-1}$.

So when we write $T \in \mathcal{T}(s_0, \theta)$, we really mean a sequence of values $v_1, \ldots, v_m, t_1, \ldots, t_{m-1}$, that is consistent with the motion commands and the environment geometry. $T(t)$ is a (position,velocity) pair which can be easily computed from the values $s_0, v_1, \ldots, v_m$ and $t_1, \ldots, t_{m-1}$. That is, we assume there is an implicit SA predicate $TR(s_0, v_1, \ldots, v_m, t_1, \ldots, t_{m-1})(t, p, v)$ which is true exactly when $T(t) = (p, v)$, where T is the trajectory determined by $s_0, v_1, \ldots, v_m, t_1, \ldots, t_{m-1}$.

4.2 Notational Conventions

Our notation closely follows Erdmann [7]. The major difference is that in the interest of compactness, rather than using separate position and velocity values p and v as Erdmann does, we mostly use (position,velocity) state pairs $s = p \times v$. Actual position, velocity and state values are denoted by lower case letters, such as p_0, v_0 and s_0. The corresponding sensor reading of a quantity is denoted with a raised asterisk. For example p_0^* denotes a sensor read-

ing of p_0, and v_0^* denotes the velocity sensor reading of the actual velocity v_0.

To take into account the uncertainty in sensor values, LMT uses uncertainty balls $B_{ep}(p)$ and $B_{ev}(v)$. $B_{ep}(p)$ is a set of positions whose distance from p under some metric is less than some distance ep, representing the possible error in the position sensor readings. Thus if p is the actual position of the robot, $B_{ep}(p)$ represents the possible readings of the position sensor. Dually, if the reading of the position sensor is p^*, then the actual position of the robot will lie in $B_{ep}(p^*)$. Analogous results hold for velocity sensor errors, where $B_{ev}(v)$ is the velocity error ball.

We add the notation s^* to mean the pair of sensor readings $p^* \times v^*$ of the actual state $s = p \times v$. And the error ball $B_e(s)$ of a state is the set $B_{ep}(p) \times B_{ev}(v)$ in state space.

We will need to make use of the set of possible states that the robot may occupy after some motion command. This is called the forward projection and is defined as:

Definition The *forward projection* of a state s_0 under the control command θ at time t is denoted $F_{s_0,\theta}(t)$. It is the set of points that the robot can reach at time t under trajectories which start at s_0 and which are consistent with the control command θ. It can be written as

$$F_{s_0,\theta}(t) = \bigcup_{T \in \mathcal{T}(s_0,\theta)} T(t) \tag{4}$$

Most of the time we will not know what the actual state s_0 of the robot is, we will only have a sensor reading s_0^* and a set of possible states R that the robot could be in. We would like to represent the set of trajectories that the robot could follow which are consistent with this information. We denote this trajectory set as $\mathcal{T}(s_0^*, \theta, R)$ and it can be defined as:

$$\begin{aligned} &T \in \mathcal{T}(s_0^*, \theta, R) \iff \\ &\exists s_0 \ (s_0 \in R \cap B_e(s_0^*)) \wedge (T \in \mathcal{T}(s_0, \theta)) \end{aligned} \tag{5}$$

Similarly, we can define a forward projection $F_{s_0^*,\theta,R}(t)$ as the set of possible states of the robot at time t which are consistent with the sensor reading s_0^*, and the fact that the initial state of the robot is contained in R.

$$\begin{aligned} &s \in F_{s_0^*,\theta,R}(t) \iff \\ &\exists s_0 \ (s_0 \in R \cap B_e(s_0^*)) \wedge (s \in F_{s_0,\theta}(t)) \end{aligned} \tag{6}$$

4.3 One-step planning: The Preimage Condition

The fundamental concept in the LMT formalism is the notion of a *preimage*. The preimage of a goal set G is the set of states from which the robot can *recognizably* reach the goal. In fine motion planning, it is not enough that the robot move through a goal region, since it may not have accurate enough sensors to determine when it is in the goal, and may halt prematurely or too late, and miss the goal.

As formalized by Erdmann, a set R is a *preimage* of a goal G under the motion command θ if and only if the following is true:

Preimage Condition For all initial sensed states s_0^* and for all trajectories $T \in \mathcal{T}(s_0^*, \theta, R)$, there is a time $t \geq 0$ such that for every set of sensor readings s^*, consistent with T at time t, the goal set G contains the set of sensor interpretations $F_{s_0^*,\theta,R}(t) \cap B_e(s^*)$.

The notion of recognizability is captured in the termination condition, which is that $F_{s_0^*,\theta,R}(t) \cap B_e(s^*) \subseteq G$. The condition for R to be a preimage of G translates into the following formula:

$$\begin{aligned} &\forall s_0^* \ \forall T \ \exists t \geq 0 \ \forall s_1^* \\ &(T \in \mathcal{T}(s_0^*, \theta, R) \ \wedge \ s_1^* \in B_e(T(t))) \Rightarrow \\ &\quad (F_{s_0^*,\theta,R}(t) \cap B_e(s_1^*) \subseteq G) \end{aligned} \tag{7}$$

We assume that the trajectories $T \in \mathcal{T}(s_0, \theta)$ are parametrizable by a polynomial number of variables, that $\mathcal{T}(s_0, \theta)$ is an SA predicate in these variables, and that there is an SA predicate to test whether $(p, v) = T(t)$. The last section gave some justification for this assumption. Now if we are given SA predicates defining the sets R and G, we can eliminate quantifiers and decide whether R is a preimage of G. But if we have a multistep plan, we somehow have to guess a sequence of intermediate subgoals G_α, and verify that each is a preimage of the next in the sequence.

The approach suggested in LMT was based on the idea of enumerating all possible preimages of the goal, then computing all preimages of those to get all possible two-step preimages etc. One way to describe all possibe preimages might be to describe a single maximal preimage which contains all others. Unfortunately, it was pointed out in [7] that maximal preimages in this sense do not always exist. So it seemed that it might truly be necessary to enumerate all subsets of state space that were preimages. This leads to computability barriers, since if one allows second order real variables which can represent subsets of $\Re^m$, the resulting algebraic theory is undecidable.

In the next section we argue that it is not necessary to consider all possible subsets of state space as preimages, only those that are consistent with some set of sensor values that might arise at run-time. This latter family of subsets can be described algebraically, and indexed by the possible sensor values, which are encoded in a finite number of real variables. Our method of finding a motion plan is both forward- and backward-chaining, in contrast to LMT, which is strictly backchaining.

5 Recognizable Sets as Subgoals

Before considering k-step plans, we consider a simple two-step plan. Let the plan consist of two commanded velocities θ_1 and θ_2. We wish to decide whether this two-step plan will guarantee successful progress into the goal region G from a given start region R. To do this we must find a suitable set of intermediate subgoals H_α such that every H_α is a preimage of G, and that starting from R, we can recognizably reach some H_α.

From the preimage equation (7) we see that to recognizably attain some subgoal H_α from R under the control command θ_1, it must be the case that

$$\begin{aligned} &\forall s_0^* \ \forall T \ \exists t_1 \geq 0 \ \forall s_1^* \ \exists H_\alpha \\ &(T \in \mathcal{T}(s_0^*, \theta_1, R) \ \wedge \ s_1^* \in B_e(T(t_1))) \Rightarrow \\ &\quad (F_{s_0^*,\theta_1,R}(t_1) \cap B_e(s_1^*) \subseteq H_\alpha) \end{aligned} \tag{8}$$

where s_0^* is the sensor reading at time 0, and s_1^* is the sensor reading at time t_1. Now preimages have the property that any subset of a preimage is also a preimage. So if H_α is a preimage and it contains $F_{s_0^*,\theta_1,R}(t_1) \cap B_e(s_1^*)$, then $F_{s_0^*,\theta_1,R}(t_1) \cap B_e(s_1^*)$ is also a preimage.

So rather than considering all H_α which are preimages of G, we need only consider those of the form $F_{s_0^*,\theta_1,R}(t_1) \cap B_e(s_1^*)$. This is sufficient because if $F_{s_0^*,\theta_1,R}(t_1) \cap B_e(s_1^*)$ is *not* a preimage of G, then there can be no larger set H_α which is a preimage.

We define the *recognizable set* $H(s_0^*, t_1, s_1^*)$ to be precisely the set of states which are consistent with the sensor readings s_0^* and s_1^* and the time t_1:

$$H(s_0^*, t_1, s_1^*) = F_{s_0^*, \theta_1, R}(t_1) \cap B_e(s_1^*) \tag{9}$$

In the formula (8), when it comes to the existential choice of the set H_α, we take the set $H(s_0^*, t_1, s_1^*)$ whenever it is a preimage of G. The formula (8) then reduces to

$$\begin{aligned} &\forall s_0^* \; \forall T \; \exists t_1 \geq 0 \; \forall s_1^* \\ &(T \in \mathcal{T}(s_0^*, \theta_1, R) \; \wedge \; s_1^* \in B_e(T(t_1))) \; \Rightarrow \; P(s_0^*, t_1, s_1^*) \end{aligned} \tag{10}$$

where $P(s_0^*, t_1, s_1^*)$ is a predicate which is true precisely when $H(s_0^*, t_1, s_1^*)$ is a preimage of G. So we next determine which of these sets are preimages of G. We use the preimage equation again, where the set R is replaced by the set $H(s_0^*, t_1, s_1^*)$, using its SA predicate computed from (9):

$$\begin{aligned} &\forall T \; \exists t_2 \geq 0 \; \forall s_2^* \\ &(T \in \mathcal{T}(s_1^*, \theta_2, H(s_0^*, t_1, s_1^*))) \; \wedge \; s_2^* \in B_e(T(t_2))) \; \Rightarrow \\ &\quad (F_{s_1^*, \theta_2, H(s_0^*, t_1, s_1^*)}(t_2) \cap B_e(s_2^*) \subseteq G) \end{aligned} \tag{11}$$

Now this is a quantified formula with free variables s_0^*, t_1 and s_1^* (and of course θ_2 which we are treating as a constant in this example). We can perform quantifier elimination on this formula to yield the quantifier-free SA predicate $P(s_0^*, t_1, s_1^*)$ which is true if and only if $H(s_0^*, t_1, s_1^*)$ is a preimage of G. We substitute the predicate $P(s_0^*, t_1, s_1^*)$ into (10), eliminate quantifiers, and we get either a true or false result depending on whether the two-step strategy θ_1, θ_2 succeeds or not.

5.1 Multi-step Plans

The methods used for two-step plans above extend naturally to multi-step plans. The general planner is both forward- and backward-chaining. It forward chains to find recognizable sets, and then backward chains to determine which recognizable sets are preimages. In this section we give the planner the added power to choose the commanded motion θ_k at execution time in response to previous sensor values. Thus θ_k becomes an exisentially quantified variable in the formulae. We could have also done this in the example of the previous section, but we omitted it in the interest of simplicity.

In the last section, we saw that the recognizable sets $H(s_0^*, t_1, s_1^*)$ are the smallest sets that can be recognizably reached after the first step. We can recursively define recognizable sets for subsequent plan steps, and they retain the property that they are the smallest sets that are recognizably reachable after, say, k plan steps. We now give a general expression for recognizable sets and make explicit their dependence on the velocity commands θ_k. In what follows, θ_k is the motion command during the k^{th} step, t_k is the time at which the k^{th} motion terminates, and s_k^* is the sensor reading at time t_k.

Definition
The *recognizable set* $H_k(s_0^*, \theta_1, t_1, s_1^*, \theta_2, \ldots, \theta_k, t_k, s_k^*)$ is a set of states recognizably reachable after k plan steps. We abbreviate it to H_k, and define it recursively as:

$$H_k = F_{s_{k-1}^*, \theta_k, H_{k-1}}(t_k) \cap B_e(s_k^*) \tag{12}$$

So for an n-step plan, we can compute k-step recognizable sets H_k using this formula for $k = 1, \ldots, n$, taking as the initial sets $H_0(s_0^*) = R \cap B_e(s_0^*)$.

Next we must decide which of the recognizable sets are preimages. Let $P_k(s_0^*, \theta_1, t_1, s_1^*, \ldots, t_k, s_k^*)$ be a predicate which is true precisely when $H_k(s_0^*, \theta_1, t_1, s_1^*, \ldots, t_k, s_k^*)$ is a preimage of G under the motion commands $\theta_{(k+1)}, \ldots, \theta_n$. Then $P_k(s_0^*, \theta_1, t_1, s_1^*, \ldots, t_k, s_k^*)$ is computed recursively by eliminating quantifiers from the right hand side of:

$$\begin{aligned} &P_k(s_0^*, \theta_1, t_1, \ldots, t_k, s_k^*) \iff \\ &\exists \theta_{k+1} \; \forall T \; \exists t_{k+1} \geq 0 \; \forall s_{k+1}^* \\ &(T \in \mathcal{T}(s_k^*, \theta_{k+1}, H_k) \; \wedge \; s_{k+1}^* \in B_e(T(t_{k+1}))) \; \Rightarrow \\ &P_{k+1}(s_0^*, \theta_1, t_1, \ldots, t_{k+1}, s_{k+1}^*) \end{aligned} \tag{13}$$

We can use this recurrence to compute preimage predicates for $k = n-1, n-2 \ldots, 0$, where the base case is the "zero-step" preimage predicate:

$$\begin{aligned} &P_n(s_0^*, \theta_1, t_1, s_1^*, \ldots, t_n, s_n^*) \iff \\ &H_n(s_0^*, \theta_1, t_1, s_1^*, \ldots, t_n, s_n^*) \subseteq G \end{aligned} \tag{14}$$

After backchaining n times, we obtain finally a predicate $P_0(s_0^*)$, which still has one free variable s_0^*. We eliminate this variable with a universal quantifier, and we obtain either a true or false value, depending on whether a guaranteed n-step motion plan does or does not exist.

So we now have a general method for finding n-step fine motion plans. We forward chain using (12) to find the recognizable sets H_k for $k = 1, \ldots, n$. The definitions above imply that all the H_k are parametric SA sets. Then we backward chain from the goal to find the preimage predicates P_k for $k = n-1, n-2 \ldots, 0$ using (13).

5.2 Termination and Choice of Next Motion Command

Our method also provides the termination predicates for the plan executor, which are precisely the predicates P_k. At the k-th step, the plan executor tests the values of t_k and s_k^* continuously, evaluating the predicate $P_k(s_0^*, \theta_1, t_1, s_1^*, \ldots, t_k, s_k^*)$ (having remembered the previous times and sensor readings). As soon as P_k returns a true value, the executor halts the motion. It has recognizably reached a multistep preimage of the goal.

We have not said so far how the plan executor chooses the next motion command θ_{k+1} if it is doing this at runtime. The formula defining P_k is (13) and we see that the first quantifier is $\exists \theta_{k+1}$. If instead of eliminating this quantifier, we leave it as a free variable, we obtain a formula $\Theta_{k+1}(s_0^*, \theta_1, t_1, \ldots, t_k, s_k^*, \theta_{k+1})$. Now all of the values of $s_0^*, \theta_1, t_1, \ldots, t_k, s_k^*$ are known to the plan executor at the end of the k^{th} step, and so it can substitute them into Θ_{k+1} to give a formula with the single free variable θ_{k+1}. Now this formula will be true for some value of θ_{k+1}, since P_k is true, and $P_k = \exists \theta_{k+1} \; \Theta_{k+1}$. So using the "sample point" procedure mentioned in the section on real algebraic algorithms, the plan executor can compute a suitable value of θ_{k+1}.

5.3 Complexity Analysis

In measuring the complexity of the algorithms we have described, we assume that the environment complexity (number of polynomials defining the environment) is m, the number of plan steps is n and the dimension of configuration space is r. When we compute the sets H_k using (12) we make use of the definition of forward projection given in (6). Using these two formulae recursively we can write down a formula for H_k directly in terms of the initial set R. This

formula has $O(kr)$ explicit existentially quantified variables $s_0, \ldots, s_{k-1}$, and there are also $O(mkr)$ "implicit" existentially quantified variables that define trajectories within the forward projection predicate. They can be eliminated to give a quantifier-free formula for H_k of size $2^{2^{O(kmr)}}$ using Collin's method.

Let $M = 2^{2^{O(nmr)}}$ be the size of the largest H_k formula. If we fully expand the recursive back-chaining formula (13) we obtain a formula with nmr quantified variables, and whose length before quantifier elimination is roughly nM. Eliminating all these quantifiers takes time $2^{2^{O(nmr)}}$ using Collins. If we retain intermediate formulae during the quantifier elimination, we obtain all the termination predicates predicates P_k and next motion command predicates Θ_k as a side effect.

This extremely high complexity seems to be unavoidable with the full generality of LMT. The alternation of existential and universal quantifiers is the source of the trouble. The alternation occurs if the plan executor is allowed to make use of sensor information to make choices about commanded motions or termination time. Of course, if the plan executor makes no choices at execution time, then it cannot make use of sensor values, and we are left with *sensorless plans* as described in [8]. Since there is only a fixed number of alternations of quantifier types in formulae for sensorless planning, plans can be found more rapidly, and the running time for a sensorless planner should be single exponential. While there is a considerable improvement in time complexity of finding sensorless plans, the loss of power is apparently severe.

6 Conclusions

We gave an algorithm for finding fine motion plans in the LMT framework. Our algorithm made use of quantifier elimination procedures for semi-algebraic predicates. Conditions for existence of a successful plan were defined as formulae with real quantified variables, and the variables were eliminated to decide if a successful plan exists. This is possible whenever the set of trajectories has a finite parametrization, and when the set of feasible trajectories $\mathcal{T}(s_0, \theta)$ is an SA subset of this parameter space. The complexity of the method was shown to grow double exponentially with the number of plan steps.

The heart of the method is its use of recognizable sets as the subgoals in multi-step plans. Our method forward chains to find all recognizable sets as parametric SA sets, and then backward chains to determine which of these sets are preimages of the goal. In the backward chaining phase, the planner produces predicates P_k which are the termination predicates for the plan executor, and Θ_k which can be used to determine the next motion command.

We observe that our methods also apply to the extension of LMT to allow *model error* [5]. Model error is variation in the geometry of the environment. If this variation is described by a finite number of parameters, then finding a guaranteed plan reduces to a classical LMT problem, where the parameters are added as extra coordinates to the configuration space. The plan executor can neither sense these parameter values, nor change them with a motion command, but these are acceptable restrictions.

References

[1] Buckley S., "Planning and Teaching Compliant Motion Strategies", MIT AI Lab TR-936, 1987.

[2] Canny J., "Some Algebraic and Geometric Computations in PSPACE", ACM Symp. Theory of Comp., Chicago, 1988.

[3] Canny J., and Reif J., "New Lower Bound Techniques for Robot Motion Planning", IEEE conf. Found. of Comp. Sci., Los Angeles, 1987.

[4] Collins G., "Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition", LNCS No. 33, Springer-Verlag, NY, pp. 135-183., 1975.

[5] Donald B., "Error Detection and Recovery for Robot Motion Planning with Uncertainty", MIT AI TR-982, 1987.

[6] Donald B., "On the Complexity of Planar Compliant Motion Planning under Uncertainty", Proc. ACM conf. on Comp. Geom., Urbana, Ill., 1988.

[7] Erdmann M., "Using Backprojections for Fine Motion Planning with Uncertainty", Int. Jour. Robotics Research **5** (1), 1986.

[8] Erdmann M., and Mason M., "An Exploration of Sensorless Manipulation", IEEE Int. Conf. Robotics and Automation, San Francisco, 1986.

[9] Grigoryev D., "Complexity of Deciding Tarski Algebra", Jour. Symb. Comp., 1988.

[10] Inoue H., "Force Feedback in Precise Assembly Tasks", MIT AI Lab memo 308, 1974.

[11] Latombe J., "Robot Motion Planning with Uncertainty: The Preimage Backchaining Approach", Tech. Rept. CS-88-1196, Stanford University, 1988.

[12] Lozano-Pérez T., Mason M., and Taylor R., "Automatic Synthesis of Fine-Motion Strategies for Robots", Int. Jour. Robotics Research **3** (1), 1984.

[13] Mason M., "Compliant Motion", Robot Motion, Brady et. al. (eds), MIT Press, Cambridge, MA, 1982.

[14] Mason M., "Automatic planning of fine motions: correctness and completeness", IEEE Conf. Robotics, Atlanta, 1984.

[15] Tarski A., "A Decision Method for Elementary Algebra and Geometry" Univ. of Calif. Press, Berkeley 2nd ed., 1951.

[16] Taylor R., Mason M., and Goldberg K., "Sensor-Based Manipulation Planning as a Game with Nature", Fourth Int. Symp. Robotics Research, Santa Cruz, 1987.