

# Computing Smooth Feedback Plans Over Cylindrical Algebraic Decompositions

Stephen R. Lindemann and Steven M. LaValle  
Department of Computer Science  
University of Illinois  
Urbana, IL 61801 USA  
{slindema, lavalle}@uiuc.edu

**Abstract**—In this paper, we construct smooth feedback plans over cylindrical algebraic decompositions. Given a cylindrical algebraic decomposition on  $\mathbb{R}^n$ , a goal state  $x_g$ , and a connectivity graph of cells reachable from the goal cell, we construct a vector field that is smooth everywhere except on a set of measure zero and the integral curves of which are smooth (i.e.,  $C^\infty$ ) and arrive at a neighborhood of the goal state in finite time. We call a vector field with these properties a smooth feedback plan. The smoothness of the integral curves guarantees that they can be followed by a system with finite acceleration inputs:  $\ddot{x} = u$ . We accomplish this by defining vector fields for each cylindrical cell and face and smoothly interpolating between them. Schwartz and Sharir showed that cylindrical algebraic decompositions can be used to solve the generalized piano movers' problem, in which multiple (possibly linked) robots described as semi-algebraic sets must travel from their initial to goal configurations without intersecting each other or a set of semi-algebraic obstacles. Since we build a vector field over the decomposition, this implies that we can obtain smooth feedback plans for the generalized piano movers' problem.

## I. INTRODUCTION

Feedback motion planning is a fundamental problem in control and robotics. If the state space is a smooth manifold  $X \subseteq \mathbb{R}^n$ , and the system satisfies the state transition  $\dot{x} = f(x, u)$ , a *feedback strategy* (also called a *control law*) is a map  $\pi : X \rightarrow \mathcal{U}$ , in which  $\mathcal{U}$  is the input space. A feedback strategy can also be seen as a vector field on  $X$ , since the choice of  $u$  at any point  $x \in X$  determines the tangent vector of the system trajectory at that point. For a feedback strategy to be useful, the behavior of the system under the strategy must have some desirable properties. For example, stability is an important consideration; the control law should prevent the system from being unbounded as time goes to infinity. Another important property is convergence to a given goal point or region. For this to be satisfied, the system should be guaranteed to converge to the goal region under application of the feedback control. In this paper, we consider feedback motion planning on the cells of a cylindrical algebraic decomposition of a bounded subset of  $\mathbb{R}^n$ . The system we consider is of the form  $\ddot{x} = u$ . If our cylindrical algebraic decomposition arises from a generalized piano movers' problem as in Schwartz and Sharir [1], with some cells of the decomposition removed because of the configuration space obstacles, then the feedback plan (vector field) we construct is guaranteed to take any initial state to the goal state while avoiding the obstacle cells, and to do so

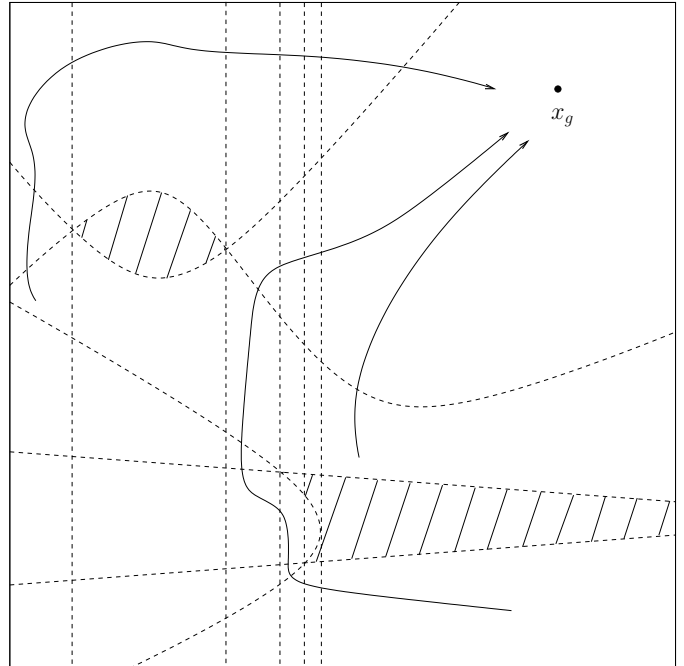


Fig. 1. The resulting cylindrical algebraic decomposition and several flows of a smooth feedback plan on the cell decomposition. The shaded regions are obstacles bounded by polynomials.

smoothly. Hence, our method computes a smooth feedback plan for the generalized piano movers' problem.

Traditional feedback control is well studied [2], but cannot be applied in many cases due to nonconvex obstacles in the environment. This is difficult enough when we are considering nonconvex obstacles in the plane; it is far more challenging when the obstacles are the semi-algebraic sets in a high-dimensional configuration space corresponding to the generalized piano movers' problem. In the algorithmic motion planning community, the solution to this has been to compute open loop trajectories linking initial and goal configurations but ignoring feedback considerations. This work is very important because it provides a way to compute trajectories for very complex, high-dimensional problems; however, it is important to think not only in terms not only of open loop trajectories, but also about feedback control.

Some have tried to make feedback more central through the construction of potential fields that have no local minima other than the goal state [3], [4]. If such a potential field can be found, the gradient of the field can be used as the velocity command for the robot. However, there are a number of difficulties associated with computing such potential fields. We will bypass these difficulties by directly constructing a vector field with the desired properties, rather than constructing a real valued function and using the gradient as the vector field. In our case, we take a cylindrical algebraic decomposition of the configuration space (which is known to be able to solve the generalized piano movers' problem [1]) and construct vector fields for each of the cells in the decomposition. We do this by inductively defining vector fields on cells of dimension one and then iteratively lifting them into more dimensions, in the same way as the decomposition itself is constructed. Different locally defined vector fields are smoothly combined using bump functions. The result is a globally defined vector field the integral curves of which are smooth and converge to a goal state. An illustration of a vector field produced by our algorithm is given in Figure 1. Our vector fields can be used directly for kinematic systems, or they can be used to develop dynamic control policies. For example, if the computed vector field is  $V(x)$ , a control policy

$$u = K(V(x) - \dot{x}) + \dot{V}(x)$$

for some feedback gain  $K$  can be used [5]. Under certain conditions, it can be shown that the system will converge to the integral curves of the constructed vector field [5], [6].

In the following section, we will review related work, focusing on how the feedback motion problem has been addressed within the robotics community and describing in detail the method of upon which ours is based. We will then briefly describe cylindrical algebraic decomposition, and our algorithm in detail. We will demonstrate that the integral curves of our vector field converge to the goal state.

## II. FEEDBACK MOTION PLANNING

### A. Background

The problem of finding a global motion plan in complex environments is difficult. Motion planning problems in robotics typically involve non-convex constraints resulting from obstacles in the environment. This presents a significant problem for traditional feedback control methods. One solution might be to use state space sampling along with dynamic programming to achieve not only feedback, but approximately optimal trajectories [7]–[9]. This may be feasible for low-dimensional spaces, but both the time- and space-complexity is exponential in the dimension of the state space, assuming that the sampling resolution remains fixed. The difficulty of feedback control for these problems motivates the development of open loop motion planning algorithms, which can at least find feasible paths through obstacle-cluttered environments. Such algorithms have been extensively studied [10], [11]. Many motion planning algorithms have been developed for kinematic systems; several, such as RRTs [12] and PDST-EXPLORE

[13] are specifically designed for systems with dynamics. Kinematic motion planning algorithms find paths which need post-processing (e.g., time-scaling [14], [15], steering [16], [17], or other transformations [18], [19]) to be transformed into trajectories for dynamical systems. In contrast, RRTs and similar planners find such trajectories directly. In either case, an open loop trajectory for the system is found. This trajectory can then be tracked using feedback.

This approach has several disadvantages, however. First, paths generated by motion planning algorithms often appear to be of poor quality, having unnecessary turns and bends in them. This may result in them being difficult to follow for a dynamical system. Second, this approach does not produce a global feedback plan, but only a local feedback plan in a neighborhood of the nominal trajectory. It would be better to solve the feedback problem once for the entire space.

Another approach, made plausible through tremendous advances in computational power, is to use motion planning algorithms themselves as the feedback mechanism. In such a model, any time the system deviated from the prescribed trajectory, the trajectory would be re-planned (probably from scratch) based on the new state of the system. This approach is extremely problematic as well. First, it has a very high computational cost, and may not be suitable for real-time applications. Second, this approach is not even guaranteed to bring the system to the goal state, although in practice it might be expected to.

These approaches, which add feedback almost as an afterthought to open loop trajectories, have significant problems, as we have seen. Consequently, there have been some attempts within the robotics community to incorporate feedback more directly. For example, the sampling-based neighborhood graph (SNG) covers the free space with balls, each of which is equipped with a local navigation function which is guaranteed to convey the robot into a ball nearer to the goal state. Other approaches to feedback motion planning in the presence of obstacles are often based on potential fields. Khatib [3] developed a method which utilized a potential field over the operational space to guide a manipulator or mobile robot to the goal. His approach suffers from local minima, however, as do many potential field methods. Waydo and Murray give a stream function method for navigation in two-dimensional environments [20]. A highly influential potential field method is that of Rimon and Koditschek [4], who show how to develop *navigation functions* (potential functions with a unique minimum at the goal and meeting certain other criteria) using potential functions in a generalized sphere world. Rimon and Koditschek have presented the most general feedback planning technique up to now; their method applies to any problem whose configuration space is topologically equivalent to a generalized sphere world. Our method is more general in that it applies to any configuration space with a well-defined cylindrical algebraic decomposition.

Finally, work by Conner *et al.* [6] computes feedback plans over cell decompositions, as does our work [21], [22]. Conner *et al.* consider an cell complex environment in  $d$ -dimensional

Euclidean space. They then impose a potential field over each individual cell, taking as the field the pullback of a potential function on a disk, which has a closed form solution. They require that the gradients of the potential fields be perpendicular to the cell boundaries, so that adjoining potential fields can be easily pieced together. Putting the individual “component control policies” together guarantees that the global control policy brings the robot to the goal. In addition to specifying a control policy for kinematic systems, they develop control policies for systems with dynamical constraints. Similarly, Lindemann and LaValle take a cell complex environment and define vector fields over the individual cells, which can also be seen as component control policies. Since this work is the primary inspiration for the current work, we describe it in detail below. Both [6] and [21] can be seen in the context of the sequential composition of funnels approach [23], in which a collection of controllers is developed, each of which converges to a goal set which is either the actual goal state or in the domain of another controller. Following a sequence of these controllers will cause the system to arrive at the goal state. This idea was further developed in [5], [24].

### B. Smooth Feedback Plans on Convex Polytopes

This work is based on our earlier work on constructing feedback plans on convex polytopes [21]. It will be useful to describe this work in some detail. The problem considered is that of smooth feedback motion planning for a point robot whose environment is a  $d$ -dimensional cell complex, each cell being a bounded  $d$ -dimensional convex polytope. Such a cell complex might be generated from a convex decomposition of a  $d$ -dimensional polygonal environment. There is a goal state  $x_g$ , and consequently a goal cell  $C_g$  containing  $x_g$ . The connectivity of the convex cells is used to construct a graph (actually, the graph is the dual of the cell complex) and a graph search algorithm such as Dijkstra’s algorithm or breadth first search is used to determine a path from each cell to  $C_g$ . Then, each cell other than  $C_g$  has a “successor” cell which is the next cell on the path to the goal cell.

Once the cell graph has been computed, we construct a vector field on each cell which has the following properties:

- 1) The vector field is smooth except for a set of measure zero and all integral curves of the vector field are smooth.
- 2) All integral curves leave the cell via the exit face, entering the designated successor of that cell.
- 3) Smoothness of the integral curves is preserved when cell boundaries are crossed.

These properties guarantee that the vector field can be used for smooth feedback motion planning for the system  $\dot{x} = u$ .

An important element of the method is the use of the fact that smooth feedback plans can be constructed using two types of simple vector fields, one per  $d$ -dimensional cell and one per  $(d - 1)$ -dimensional face, blended together using bump functions. Since bump functions smoothly interpolate between two functions (more generally, partitions of unity are capable of smoothly blending arbitrarily many functions together),

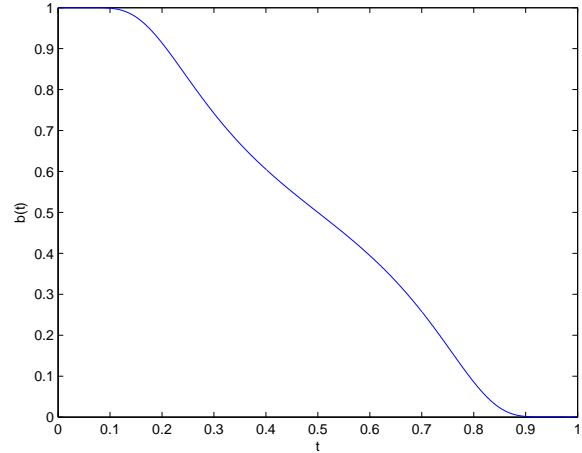


Fig. 2. A bump function: if  $\lambda(s) = (1/s)e^{-1/s}$ , then  $b(s) = 0$  for  $s \leq 0$ ,  $b(s) = 1$  for  $s \geq 1$ , and  $b(s) = \frac{\lambda(s)}{\lambda(s) + \lambda(1-s)}$  for  $0 < s < 1$ .

the system can transition from following one component vector field to another without any loss of smoothness. Bump functions are defined as follows:

**Definition 1** Let  $X$  be a smooth manifold, and let  $K$  be a closed set and  $U$  an open set,  $K \subset U \subseteq X$ . A bump function over  $U$  is a smooth, real valued function  $\rho : X \rightarrow [0, 1]$  such that:

- 1)  $\rho$  has support contained in  $U$ .
- 2)  $\rho(x) = 1$  for every  $x \in K$ .

Bump functions are in general difficult to find, even though their existence is guaranteed. However, it is simple to construct a bump function on the real line; an illustration of this bump function is given in Figure 2. The bump function has the important property that all derivatives equal zero at the endpoints of the unit interval.

In order for the approach to work, one more piece must be put in place. Within each cell, the vector fields must be blended in such a way so that on each face, the resulting vector field is identically equal to the face vector field. In the interior of each cell, all face vector fields must be smoothly interpolated between. We use the cell vector field (also called the *attractor* vector field) as an intermediary between the different face vector fields, interpolating between them. This is done using the interior generalized Voronoi diagram (GVD) of the cell. The GVD partitions the cell into regions corresponding to each face; in each region, bump functions are used to blend between the face vector field and the attractor vector field. On the GVD itself, the vector field is identically equal to the attractor vector field, guaranteeing that all face vector fields are smoothly blended together in the interior of the cell. In order to use the bump function to blend between the face vector field and the attractor vector field, the parameter of the bump function must be constructed in such a way that it will be equal to one on any of the faces of the GVD and zero on

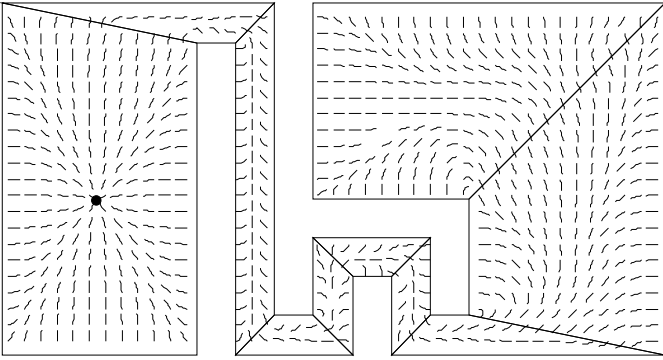


Fig. 3. A smooth feedback plan in a two-dimensional environment.

the face of the cell. We use a product of analytic switches to do this. Formally, for any point  $p$  it is defined as

$$s(p) = 1 - \prod_{j \neq i} \frac{d(p, f_j) - d(p, f_i)}{d(p, f_j)}. \quad (1)$$

in which  $\{f_i\}_1^n$  are the faces of the GVD and  $d(p, f_i)$  is the distance from  $p$  to face  $f_i$ . This function is smooth (except at the vertices of the cell), and has the desired property of being identically equal to one on the exit face and zero on all other boundaries.

Since the bump function smoothly blends the face and attractor vector fields together, a vector field is obtained which is smooth over the entire cell. With small modifications, the above approach can be used in the goal cell as well. Piecing the individual cells together results in a vector field which is smooth over the entire free space. Normalize the vector field at every point, the global vector field  $V(p)$  is defined as  $V(p) = \text{norm}(b(p)V_f(p) + (1 - b(p))V_a(p))$ , in which  $V_f$  is the face vector field for that point,  $V_a$  the attractor vector field,  $b(p) = b(s(p))$  is the bump function, and  $\text{norm}$  is the normalization function. An example of the trajectories produced by the method is given in Figure 3.

### III. CYLINDRICAL ALGEBRAIC DECOMPOSITION

Cylindrical algebraic decomposition was used to solve the piano movers' problem first by Schwartz and Sharir [1]. They used the Collins decomposition [25] to partition the configuration space into free and obstacle cells and demonstrated how to find a path from a point in one free cell to a point in any other connected free cell. Cylindrical algebraic decomposition (abbreviated CAD) is extremely powerful; in fact, it can be used to solve the first-order theory of the reals [?]. Given this, it is not surprising that it can solve the generalized piano movers' problem as well.

A *cylindrical algebraic decomposition* (CAD) of  $\mathbb{R}^n$  is defined in the following inductive way (see [26] for a more formal definition):

#### Definition 2

- 1) A cylindrical algebraic cell  $C_1$  in dimension one is either an interval  $(a, b)$  or a point  $a$ .
- 2) A cell  $C_n$  in dimension  $n$  has one of the two forms: it is either the set of pairs  $\{(x, y) : x \in C_{n-1}, f(x) < y < g(x)\}$  or the set of pairs  $\{(x, y) : x \in C_{n-1}, y = f(x)\}$ , in which  $f, g : \mathbb{R}^{n-1} \rightarrow \mathbb{R}$ .

The cells' cylindrical structure is apparent from the definition. For a set of polynomials  $\mathcal{P}$  taken from the set  $\mathbb{Q}[x_1, \dots, x_n]$  (polynomials over the field of rational numbers  $\mathbb{Q}$ ), a CAD adapted to  $\mathcal{P}$  is one in which each cell in the decomposition is sign-invariant under  $\mathcal{P}$ . The number of cells in the decomposition is doubly exponential in the dimension of the space [25].

In addition to proposing the decomposition, Collins gave an algorithm to compute it. This algorithm (which we will refer to as the CAD algorithm) has two phases. In the first phase, the polynomials of  $\mathcal{P}$  are projected down one dimension at a time, using the canonical projection. Once the polynomials have been projected into  $\mathbb{R}^1$ , the critical points are located; these points, and the corresponding open intervals, become the cells of  $\mathcal{L}_1$ . Then in the second phase, the cells of  $\mathcal{L}_1$  are lifted into  $\mathbb{R}^2$ , becoming cylinders partitioned based on the critical points of the polynomials which are now in  $\mathbb{Q}[x_1, x_2]$ . This is repeated, each time lifting up and partitioning the resulting cylinders, until  $\mathbb{R}^n$  is reached. At that point, a sign-invariant partition of  $\mathbb{R}^n$  has been obtained. As noted in [1], [26], the unbounded cells can be treated as the others by considering the set of polynomials to include  $x_i = \pm\infty$ , for  $i = 1, \dots, n$ . More details can be found in [?], [26], [27]. A (very) simple illustration can be seen in Figure 4.

Schwartz and Sharir show how to use the CAD algorithm to solve the generalized piano movers' problem. If the collision constraints on the robot are semi-algebraic in the configuration space, then each cell in the resulting CAD of the configuration space is either completely in collision or completely collision-free. If a graph representing the connectivity of the CAD cells is constructed, then it can be searched to find a collision-free cell path from the cell containing the initial state to the one containing the goal state, if one exists. The only remaining step is to specify a continuous path from the initial to the goal state which goes from cell to cell in the solution cell path without entering any other cells. Schwartz and Sharir show how this can be done. Note that using the standard convention that the free configuration space is open, the robot is guaranteed to move from one full-dimensional cell to another, through a connecting cell of one dimension lower. Hence only these cells need to be considered to determine connectivity. In order to determine the connectivity relations efficiently, Schwartz and Sharir make a stronger assumption on the set of polynomials  $\mathcal{P}$  than is required for the basic CAD algorithm. The assumption of "well-basedness" eliminates local pathology, but local connectivity can still be quite complicated. See [1] for more details.

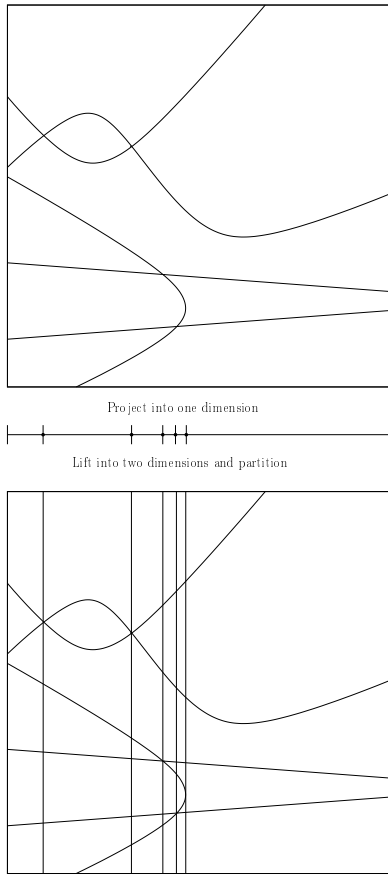


Fig. 4. The polynomials corresponding to the obstacles from Figure 1, and the steps of the CAD algorithm.

#### IV. SMOOTH FEEDBACK PLANS ON CYLINDRICAL CELLS

Now we will proceed to describe our algorithm for generating smooth feedback plans over CADs. We will construct smooth feedback over individual cells and then guarantee that smoothness is preserved across boundaries crossed by the resulting integral curves. We assume that the input to our algorithm is the entire cylindrical algebraic decomposition, consisting of all cells of level  $n$  and their corresponding algebraic descriptions, and a connectivity graph corresponding to the connectivity of the  $n$ -dimensional cells in the decomposition. In the construction of a CAD, it is possible to generate a point in each cell; these are called *algebraic points*, and we assume that we are given these as well (note that it is trivial to compute algebraic points given the full algebraic descriptions of each cell). The connectivity graph can be searched to determine the cell path to the goal cell from any cell in the connected component of the goal; this determines the successor of each cell. We will construct a vector field over the closure of each cell such that all integral curves are guaranteed to reach the face between the cell and its successor, without reaching any other face. We require all integral curves to be smooth, and smoothness must be preserved across the faces separating a cell from its successor. We will discuss our

algorithm in terms of an  $n$ -dimensional cell  $C$  (and its closure  $\bar{C}$ ) and its successor  $S$ , both full-dimensional cells of level  $n$ . There is an  $(n-1)$ -dimensional face  $F_S$  which connects them.

We know that  $C$  is bounded by upper and lower bounding polynomials in each dimension; let  $u_i$  and  $l_i$  be these polynomials in dimension  $i$ . It is intuitive that  $C$  should have as many faces as bounding polynomials, but this is not actually true. This is the case because when adjacent cells in  $i$  dimensions are lifted into  $\mathbb{R}^{i+1}$ , a full-dimensional cell of level  $i+1$  can have multiple neighboring cells in the adjacent cylinder (arbitrarily many, in fact). On the other end of the spectrum, a cell may have no faces corresponding to a particular dimension because of certain polynomial intersections. Denote by  $\bar{F}_i^+$  the union of all upper bounding faces of  $C$  corresponding to dimension  $i$ , and by  $\bar{F}_i^-$  the union of all lower bounding faces of  $C$  corresponding to dimension  $i$ . Note that  $\bar{F}_i^+$  corresponds to  $u_i = 0$  and  $\bar{F}_i^-$  corresponds to  $l_i = 0$ . Then if  $F_S$  is an upper face corresponding to dimension  $i$ , for example, we have that  $F_S \subseteq \bar{F}_i^+$ , but it can be the case that  $F_S$  is a “window” in  $\bar{F}_i^+$ . We know that the bounding polynomials of  $F_S$  always correspond to the bounding polynomials of either  $C$  or  $S$ ; windows are formed when they correspond to the bounding polynomials of  $S$  rather than those of  $C$ . If they all corresponded to the bounding polynomials of  $C$ , then we have  $F_S = \bar{F}_i^+$  and no window would exist.

We will construct a vector field over  $\bar{C}$  similar to our approach in [21]. We will define appropriate smooth distance functions representing the distance to each face of the cell, as well as face vector fields for each face and an attractor vector field for the cell. Face vector fields are easily defined; any face  $F \subseteq \bar{F}_i^+$  will be assigned a vector field of  $-\frac{\partial}{\partial x_i}$  and any face  $F \subseteq \bar{F}_i^-$  will be assigned a vector field of  $+\frac{\partial}{\partial x_i}$ . This is the case except for  $F_S$ ; for any  $x \in F_S$ , the face vector field  $V_f$  is defined as

$$V_f(x) = \begin{cases} +\frac{\partial}{\partial x_i} & \text{if } F_S \subseteq \bar{F}_i^+ \\ -\frac{\partial}{\partial x_i} & \text{if } F_S \subseteq \bar{F}_i^- \end{cases}$$

These face vector fields guarantee that the integral curves will not cross any face except the exit face, where they cross from  $C$  to its successor,  $S$ .

Recall that through the construction of the CAD, algebraic points have been computed which lie in the interior of each cell. Denote the algebraic point in  $F_S$  as  $p_a$ . Also, define *relative height* as follows: for any point  $x = (x_1, \dots, x_n) \in \bar{C}$  and dimension  $i$ , define  $h_i : \bar{C} \rightarrow [0, 1]$  as

$$h_i(x) = \frac{x_i - l_i(x)}{u_i(x) - l_i(x)}. \quad (2)$$

This is a smooth mapping, since both the upper and lower bounding polynomials are smooth. For the sake of completeness, we can arbitrarily define  $h_i(x)$  to be zero if  $u_i(x) = l_i(x)$ ; this can only occur when faces corresponding to certain dimensions are missing, and will not affect our algorithm at all, because these faces will always be less than  $(n-1)$ -dimensional and are consequently irrelevant to full-dimensional cell connectivity. We define the relative coordi-

nates of  $x$  as  $h(x) = (h_1(x), \dots, h_n(x)) \in [0, 1]^n$ . Then, we can intuitively define the attractor vector field  $V_a$  as that which induces a straight line path toward  $p_a$ , when considered in relative coordinates. Formally, this can be computed using the Jacobian of  $h$ , which is guaranteed to have full rank since  $h$  is a diffeomorphism on  $C$ :  $V_a(x) := (Jh(x))^{-1}(h(p_a) - h(x))$ .

Now, all we need is to define acceptable distance functions to each face. Then, we can blend the component vector fields together as in the polygonal case, and we will show that the integral curves of the resulting vector field always reach the goal. The distance function is easy to define, using the relative height functions. Define the scaled perpendicular distance function  $d_{\perp}$  as follows:

$$d_{\perp}(x, F) = \begin{cases} \frac{1-h_i(x)}{1-h_i(p_a)} & \text{if } F \subseteq \bar{F}_i^+ \\ \frac{h_i(x)}{h_i(p_a)} & \text{if } F \subseteq \bar{F}_i^- \end{cases} \quad (3)$$

As required, the distance function equals zero on the face itself and is greater than zero elsewhere inside the cell. Note that it multiple faces which correspond to the same bounding polynomial will have the same distance; this is acceptable, because such faces will have the same face vector field. This holds except for all faces except the face  $\bar{F}_i^{\pm}$  such that  $F_S \subseteq \bar{F}_i^{\pm}$  (we use the notation  $\bar{F}_i^{\pm}$  to indicate that it can be either the upper or lower bounding face corresponding to dimension  $i$ ). We will need to define another distance function to use to distinguish  $F_S$  from the remainder of  $\bar{F}_i^{\pm}$ , to use when the point  $x$  is in the region of the cell closest to  $\bar{F}_i^{\pm}$ .

Understanding cell connectivity is important for computing the distance to  $F_S$ , because  $F_S$  can be a window in the larger face  $\bar{F}_i^{\pm}$ , as discussed above. It is useful to note that if  $F_S \subseteq \bar{F}_i^{\pm}$ , then both  $C$  and  $S$  were lifted from the same full-dimensional cell in  $L_{i-1}$ . This means that the ‘‘parents’’ of  $C$  and  $S$  in  $L_i$  were adjacent cells in the same cylinder, separated by an  $(i-1)$ -dimensional face. The parent cells shared a complete face at that level; lifting the cells into higher dimensions may have restricted the area of the face which they share until only a window remains.

Keeping in mind that each successive lifted dimension adds constraints which restrict the shared face between  $C$  and  $S$ , consider some point  $x \in \bar{F}_i^{\pm}$ . It is simple to verify whether or not  $x$  lies in  $F_S$  (simply check to see if it satisfies the constraints of the bounding polynomials of  $F_S$ ). In addition to this, we need a smooth function defined over all of  $\bar{F}_i^{\pm}$  that can serve as a distance function, indicating how far  $x$  is from  $F_S$  even if  $x \notin F_S$ . One option which seems obvious but which is incorrect would be to compute the distance to each of the bounding polynomials of  $F_S$  which is unsatisfied, smooth them using a bump function if necessary, and add them together. This is incorrect because the bounding polynomials of  $F_S$  are not necessarily well-defined for any point  $x \in \bar{F}_i^{\pm}$ . The bounding polynomials  $u_2^F$  and  $l_2^F$  are not well-defined unless  $u_1^F$  and  $l_1^F$  are satisfied. Similarly, the bounding polynomials  $u_j^F$  and  $l_j^F$  are not guaranteed to be well-defined unless  $u_k^F$  and  $l_k^F$  are satisfied, for all  $k \in i+1, \dots, j-1$ ; this happens when the bounding polynomials of  $F_S$  coincide with

those of  $S$  rather than those of  $C$ . Consequently, our distance function will only depend on  $u_j^F$  and  $l_j^F$  if all lower bounding polynomials are satisfied.

We will construct a function which is uniformly equal to one outside  $F_S$ , uniformly zero on some subset of  $F_S'$ , smoothly transitioning between the two on  $F_S \setminus F_S'$ . We need to make several definitions in order to construct this function. First, define  $z_j^+(x)$  and  $z_j^-(x)$  as the zeros of  $u_j^F$  and  $l_j^F$  that correspond to  $x$ : namely,  $z_j^+(x)$  and  $z_j^-(x)$  are identical to  $x$  in all coordinates except for coordinate  $j$ , which is chosen so that  $u_j^F(z_j^+(x)) = l_j^F(z_j^-(x)) = 0$ . Then, for some  $\alpha \in (0, 1)$  define the *satisfaction function*  $w_j : \bar{F}_i^{\pm} \rightarrow [0, 1]$  as

$$w_j(x) = b \left( \frac{1}{\alpha} \left( \frac{|h(x) - h(p_a)|}{|h(z_j^+(x)) - h(p_a)|} - (1 - \alpha) \right) \right) + b \left( \frac{1}{\alpha} \left( \frac{|h(x) - h(p_a)|}{|h(z_j^-(x)) - h(p_a)|} - (1 - \alpha) \right) \right). \quad (4)$$

Then define the distance function  $\hat{d}_j$  as follows:

$$\begin{aligned} \hat{d}_{i+1}(x) &= w_{i+1}(x) \\ &\vdots \\ \hat{d}_j(x) &= \hat{d}_{j-1}(x) + (1 - \hat{d}_{j-1})w_j(x) \end{aligned} \quad (5)$$

The satisfaction function  $w_j$  considers the bounding polynomials corresponding to dimension  $j$  and is identically one for points above the upper bounding polynomial or below the lower bounding polynomial in that dimension. It equals zero for any  $x$  such that the difference in relative height from  $x$  to  $p_a$  (in direction  $x_j$ ) is less than  $(1 - \alpha)$  times the difference in relative height from  $p_a$  to the boundary of  $F_S$ , again in direction  $x_j$ . These can be used to construct the final distance function  $\hat{d}_n$ , which for any point  $x \in \bar{F}_i^{\pm}$  indicates the ‘‘distance’’ from that point to  $F_S$ , and does so smoothly. The important results are summarized in the proposition below:

**Proposition 1** *The following properties hold:*

- 1) For all  $j$ ,  $\hat{d}_j$  is well-defined.
- 2) The function  $\hat{d}_n$  is smooth, identically equal to one on  $\hat{F}_i^{\pm} \setminus F_S$ , and identically equal to zero on an open subset of  $F_S$ .

*Proof:* We prove the first property by induction. As we have already indicated,  $w_j(x)$  is only guaranteed to be well-defined if the polynomial constraints  $l_k^F$  and  $u_k^F$  are satisfied for all  $k \in i+1, \dots, j-1$ . For  $1 \leq k \leq i$ , the constraints are always satisfied because the cells  $C$  and  $S$  are in the same cylinder in the projection into  $\mathbb{R}^i$ . Therefore, we know that the base case  $\hat{d}_{i+1}$  is well-defined. Now assume that  $\hat{d}_j$  is well-defined and consider  $\hat{d}_{j+1}$ . The function  $\hat{d}_{j+1}$  will be well-defined if  $\hat{d}_j = 1$  for any point  $x$  such that  $w_{j+1}$  is not well-defined (since the term containing  $w_{j+1}$  will then vanish). But this fact is apparent from the definition of  $\hat{d}_j$ ; if some constraint  $l_k^F$  or  $u_k^F$  is not satisfied, then we have  $\hat{d}_l = 1$  for all  $k \leq l \leq j$ . Therefore  $\hat{d}_j = 1$  over any point where

$w_{j+1}$  is not well-defined, and so  $\hat{d}_{j+1}$  is well-defined over all of  $\bar{F}_i^\pm$ .

For the second property, the above proof also yields the fact that  $\hat{d}_n$  is identically equal to zero on  $\bar{F}_i^\pm \setminus F_S$ . It is also readily apparent that if all polynomial constraints are satisfied by a factor of  $(1 - \alpha)$ , then we have  $\hat{d}_n = 0$ . So we simply need to verify that  $\hat{d}_n$  is smooth. It is constructed using smooth functions, so all we need to verify is that the derivatives exist on the constraint polynomials, which is the boundary where the satisfaction functions become ill-defined. This can be argued inductively, as above. The base case,  $\hat{d}_{i+1}$ , is clearly smooth. Now assume that  $\hat{d}_j$  is smooth. Just as guaranteeing that  $\hat{d}_j = 1$  wherever  $w_{j+1}$  is not well-defined is sufficient to make  $\hat{d}_{j+1}$  well-defined, we use the property that all derivatives of the bump function  $b(s)$  are zero outside the unit interval. This implies that anywhere the function  $w_{j+1}$  is not well-defined, the derivatives of  $\hat{d}_j$  all equal zero. Consequently, all derivatives of  $\hat{d}_{j+1}$  exist and are well-defined over  $\bar{F}_i^\pm$ , and the function  $\hat{d}_n$  is smooth. ■

Using these distance functions, for any point  $x \in C$  we can determine the face in whose region of influence it lies (i.e., which face it is “closest to”). There are three different cases. Assume as before that  $F_S \subseteq \bar{F}_i^\pm$ . First, for some face  $\bar{F}_j^\pm$  with  $j \neq i$ , we say that  $x$  lies in the region of influence of  $\bar{F}_j^\pm$  if  $d(x, \bar{F}_j^\pm) \leq d(x, \bar{F}_k^\pm)$ , for all  $k$ . Second, we say that  $x$  lies in the region of influence of  $F_S$  if  $d(x, \bar{F}_i^\pm) \leq d(x, \bar{F}_k^\pm)$  for all  $k$  and if  $\hat{d}_n(x) \leq 1 - \hat{d}_n(x)$ . Finally,  $x$  lies in the region of influence of  $\bar{F}_i^\pm \setminus F_S$  if  $\hat{d}_n(x) \leq 1 - \hat{d}_n(x)$  for all  $k$  and if  $1 - \hat{d}_n(x) \leq \hat{d}_n(x)$ .

The final step is to define a function for each face which interpolates between a value of zero on the face itself and a value of one on the boundaries of its region of attraction (loosely, the “faces” of the GVD). As in our previous work, we use a product of analytic switches. For any face  $\bar{F}_j^\pm$  with  $j \neq i$ , use the following:

$$s(p) = \prod_{\bar{F} \neq \bar{F}_j^\pm} \frac{d_\perp(p, \bar{F}) - d_\perp(p, \bar{F}_j^\pm)}{d_\perp(p, \bar{F})}, \quad (6)$$

in which  $\bar{F} \in \mathcal{F}$  are the faces of  $C$ . This function is smooth (except where faces meet), and has the desired property of being identically equal to one on the face of the cell and zero on the boundary of the region of influence. Then, using the shorthand  $b(p) = b(s(p))$ , we define the global vector field  $V$  at point  $p$  as  $V(p) = \text{norm}(b(p)V_f(p) + (1 - b(p))V_a(p))$ , in which  $V_f$  is the face vector field for the face in whose region of influence  $p$  lies,  $V_a$  the attractor vector field,  $b$  the bump function, and  $\text{norm}$  is the normalization function, so that  $V$  is a unit vector field.

We must also define the vector field on the goal cell so that the integral curves converge to the goal point  $x_g$  inside the goal cell. All face vector fields point inward in this case; the attractor vector field is the vector that points from  $x$  to  $x_g$ , in relative coordinates. As before, this is defined as  $V_a(x) := (Jh(x))^{-1}(h(x_g) - h(x))$ . Similarly, the  $d_\perp$  function should

be modified to consider coordinates relative to the goal point  $x_g$  rather than  $p_a$ .

### A. Theoretical Results

Now, we need to establish that the feedback plan associated with our constructed vector field has all the properties we require.

**Theorem 1** *The vector field  $V$  is smooth except for a set of measure zero and has smooth integral curves.*

*Proof:* Consider the functions used in the construction of  $V$  in a particular cell. The perpendicular distance function  $d_\perp$  is smooth since the bounding polynomials of the cell are smooth, and the satisfaction functions and distance functions  $\hat{d}_j$  are likewise smooth. The parameter function  $s$  is smooth except on the  $(n - 2)$  dimensional surfaces where faces meet, and the integral curves never go through these places. As we know, the bump functions are smooth. They guarantee smoothness across cell boundaries and between regions of influence within a cell because all derivatives equal zero there (this is easily verified). Hence  $V$  is smooth except on  $(n - 2)$  dimensional sets, and the integral curves are all smooth. ■

**Theorem 2** *The integral curves of  $V$  never lead to obstacle collision.*

*Proof:* This property is obvious from the construction of the vector field. In any collision free cell, the face vector field corresponding to an obstacle face will be inward pointing, because an obstacle face can never be the exit face. The vector field  $V$  is identically equal to the face vector field on the face itself, due to the bump function and its parameter function. Hence, the vector field always points away from obstacle faces and the integral curves never lead to collision. ■

**Theorem 3** *The integral curves of  $V$  converge to the goal.*

*Proof:* First, we show that for any non-goal cell  $C$ , all the integral curves of  $C$  reach the exit face  $F_S$  and thus enter the successor cell  $S$ . Recall that the attractor vector field is defined as  $V_a(x) := (Jh(x))^{-1}(h(p_a) - h(x))$ , in which  $p_a$  is the algebraic point in the exit face  $F_S$ . Hence following the integral curves of this vector field will cause the relative coordinates to converge to those of  $p_a$ : namely,  $h_j(x) \rightarrow h_j(p_a)$ ,  $1 \leq j \leq n$ . The face vector fields corresponding to all  $\bar{F}_j^\pm$ ,  $j \neq i$  also cause the relative coordinates to converge. The only exception is  $h_i$ , which has to be considered separately because the vector field corresponding to  $\bar{F}_i^\pm \setminus F_S$  points away from  $p_a$ . Consider all dimensions except dimension  $i$ . We know that the relative coordinates will converge to a neighborhood of those of  $p_a$  in some finite time  $T$  (again, not considering dimension  $i$ ). This implies that for a suitably chosen neighborhood, the region of influence of  $\bar{F}_i^\pm \setminus F_S$  cannot be entered after time  $T$ , because it lies entirely outside this neighborhood. Consequently, we can guarantee the convergence of  $h_i$  after time  $T$ , and all relative coordinates are guaranteed to converge. Once within a

neighborhood of  $p_a$  (in relative coordinates) in all dimensions, it is simple to observe that the integral curves reach the exit face  $F_S$  in finite time, since the face vector field of  $F_S$  is outward-pointing.

The case of the goal cell is similar. In this case, the attractor vector field and face vector fields all cause the relative coordinates to converge to those of the goal point. Therefore, for any neighborhood of the goal point, the integral curves will converge in finite time. Since the integral curves of  $V$  reach the exit face of any cell in finite time, and reach the goal point from any face of the goal cell in finite time, we have the global result that all integral curves of  $V$  reach a neighborhood of the goal state in finite time. ■

## V. CONCLUSIONS

In conclusion, we have introduced an algorithm for constructing a vector field on the cells of a cylindrical algebraic decomposition. Since CAD algorithms solve very general motion planning problems [1], [11], this implies that we can provide smooth feedback plans for these problems as well. To the authors' knowledge, this is the first construction of smooth feedback plans with this level of generality. It is also of interest that although a certain amount of detail is required to guarantee smoothness, the cylindrical structure of the cells lends itself quite readily to the construction of vector fields with smooth integral curves.

Due to the complexity of cylindrical algebraic decompositions, it is highly unlikely that this method will be implemented and used in its full generality. However, it has more than purely theoretical interest. It may be reasonable to apply to feedback planning for the rod [28] or for polygonal robots translating and rotating in the plane [29]. Additionally, these ideas can be applied to other specialized cell decompositions, or used in conjunction with a precomputed path to provide feedback in a neighborhood of the path. In the future, we plan to explore these avenues of research.

## ACKNOWLEDGMENTS

This work was funded in part by NSF Awards 9875304, 0118146, and 0208891.

## REFERENCES

- [1] J. T. Schwartz and M. Sharir, "On the Piano Movers' Problem: II. General techniques for computing topological properties of algebraic manifolds," *Communications on Pure and Applied Mathematics*, vol. 36, pp. 345–398, 1983.
- [2] C.-T. Chen, *Linear System Theory and Design*. New York: Holt, Rinehart, and Winston, 1984.
- [3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.
- [4] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential fields," *IEEE Transactions on Robotics & Automation*, vol. 8, no. 5, pp. 501–518, Oct. 1992.
- [5] A. A. Rizzi, "Hybrid control as a method for robot motion programming," in *Proceedings IEEE International Conference on Robotics & Automation*, 1998, pp. 832–837.

- [6] D. C. Conner, A. A. Rizzi, and H. Choset, "Composition of local potential functions for global robot control and navigation," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003, pp. 3546–3551.
- [7] D. Bertsekas, *Dynamic Programming and Optimal Control: Volume I*. Belmont, MA, USA: Athena Scientific, 2000.
- [8] S. M. LaValle and P. Konkimalla, "Algorithms for computing numerical optimal feedback motion strategies," *International Journal of Robotics Research*, vol. 20, no. 9, pp. 729–752, Sept. 2001.
- [9] J. N. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1528–1538, Sept. 1995.
- [10] J.-C. Latombe, *Robot Motion Planning*. Boston, MA: Kluwer, 1991.
- [11] S. M. LaValle, *Planning Algorithms*. Cambridge University Press (also available at <http://msl.cs.uiuc.edu/planning/>), to be published in 2006.
- [12] S. M. LaValle and J. J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, B. R. Donald, K. M. Lynch, and D. Rus, Eds. Wellesley, MA: A K Peters, 2001, pp. 293–308.
- [13] A. Ladd and L. E. Kavraki, "Fast exploration for robots with dynamics," in *Proceedings Workshop on Algorithmic Foundations of Robotics*, Zeist, The Netherlands, July 2004.
- [14] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *International Journal of Robotics Research*, vol. 4, no. 3, pp. 3–17, 1985.
- [15] K. G. Shin and N. D. McKay, "Minimum-time control of robot manipulators with geometric path constraints," *IEEE Transactions on Automatic Control*, vol. 30, no. 6, pp. 531–541, 1985.
- [16] F. Lamiroux and J.-P. Laumond, "Flatness and small-time controllability of multibody mobile robots: Application to motion planning," *IEEE Transactions on Automatic Control*, vol. 45, no. 10, pp. 1878–1881, Apr. 2000.
- [17] R. M. Murray and S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *IEEE Transactions on Automatic Control*, vol. 38, no. 5, pp. 700–716, 1993.
- [18] J.-P. Laumond, S. Sekhavat, and F. Lamiroux, "Guidelines in nonholonomic motion planning for mobile robots," in *Robot Motion Planning and Control*, J.-P. Laumond, Ed. Berlin: Springer-Verlag, 1998, pp. 1–53.
- [19] S. Sekhavat, P. Svestka, J.-P. Laumond, and M. H. Overmars, "Multilevel path planning for nonholonomic robots using semiholonomic subsystems," *International Journal of Robotics Research*, vol. 17, pp. 840–857, 1998.
- [20] S. Waydo and R. M. Murray, "Vehicle motion planning using stream functions," in *IEEE Int. Conf. Robot. & Autom.*, 2003, pp. 2484–2491.
- [21] S. R. Lindemann and S. M. LaValle, "Smoothly blending vector fields for global robot navigation," in *Proceedings IEEE Conference Decision & Control*, 2005, pp. 3353–3359.
- [22] S. R. Lindemann, I. I. Hussein, and S. M. LaValle, "Real time feedback control for nonholonomic mobile robots with obstacles," in *Proceedings IEEE Conference on Decision & Control*, 2006.
- [23] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *International Journal of Robotics Research*, vol. 3, no. 1, pp. 3–24, 1984.
- [24] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *International Journal of Robotics Research*, vol. 18, no. 6, pp. 534–555, 1999.
- [25] G. E. Collins, "Quantifier elimination for real closed fields by cylindrical algebraic decomposition," in *Proceedings Second GI Conference on Automata Theory and Formal Languages*. Berlin: Springer-Verlag, 1975, pp. 134–183, lecture Notes in Computer Science, 33.
- [26] S. Basu, R. Pollack, and M.-F. Roy, *Algorithms in Real Algebraic Geometry*. Berlin: Springer-Verlag, 2003.
- [27] S. M. LaValle, *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006, available at <http://planning.cs.uiuc.edu/>.
- [28] J. Bañon, "Implementation and extension of the ladder algorithm," in *Proceedings IEEE International Conference on Robotics & Automation*, 1990, pp. 1548–1553.
- [29] F. Avnaim, J.-D. Boissonnat, and B. Faverjon, "A practical exact planning algorithm for polygonal objects amidst polygonal obstacles," in *Proceedings IEEE International Conference on Robotics & Automation*, 1988, pp. 1656–1660.