Simple and Efficient Algorithms for Computing Smooth, Collision-Free Feedback Laws Over Given Cell Decompositions

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

***Abstract*— This paper presents a novel approach to computing feedback laws in the presence of obstacles. Instead of computing a trajectory between a pair of initial and goal states, our algorithms compute a vector field over the entire state space; all trajectories obtained from following this vector field are guaranteed to asymptotically reach the goal state. As a result, the vector field globally solves the navigation problem and provides robustness to disturbances in sensing and control. The vector field's integral curves (system trajectories) are guaranteed to avoid obstacles and are C^∞ smooth. We construct a vector field with these properties by partitioning the space into simple cells, defining local vector fields for each cell, and smoothly interpolating between them to obtain a global vector field. We present an algorithm that computes these feedback controls for a kinematic point robot in an arbitrary dimensional space with piecewise linear boundary; the algorithm requires minimal preprocessing of the environment and is extremely fast during execution. For many practical applications in two-dimensional environments, full computation can be done in milliseconds. We also present an algorithm for computing feedback laws over cylindrical algebraic decompositions, thereby solving a smooth feedback version of the generalized piano movers' problem.**

I. Introduction

Motion planning and feedback control are fundamental and well-studied problems in robotics. They both address the problem of motion: how to control a system so that it safely arrives at a target location. A feedback law is a vector field which determines which input to apply from any state; the trajectories described by the vector field all converge to the goal state. Feedback control techniques have been studied extensively for environments without obstacles, and they provide robust and effective solutions to the global navigation problem in that setting. Nearly all practical robotics problems, however, involve obstacles. Robots are required to operate in environments that are cluttered with obstacles, and must solve tasks while avoiding them. Since traditional feedback control methods do not take obstacles into account, their usefulness for these problems is limited.

Traditional motion planning approaches, on the other hand, use geometric algorithms to achieve obstacle avoidance. However, they generally only compute individual trajectories between given initial and goal states. The computed trajectories are *open loop*: that is, they are defined as a function of time only, which leaves the robot no way to recover if it deviates from the prescribed path during execution. Feedback controllers, on the other hand, are much more robust. Because they are defined as vector fields defined over the entire state space, there is no need to be concerned with tracking a particular path; the robot can simply follow the controller from its current state and be guaranteed to reach the goal state. The fact that the current state of the system is used to determine the input to apply makes this type of control *closed loop*.

Ideally, it is desirable to combine the robustness of feedback with the ability of combinatorial algorithms to efficiently take obstacles into account. The feedback law should provide global asymptotic convergence to the goal state while guaranteeing obstacle avoidance. For any state, the value of the feedback law should be easy to compute. The system trajectories obtained from following the feedback law should be smooth, in contrast to the jagged paths that motion planning algorithms often produce. The most popular way to address this problem (sometimes termed *simultaneous planning and control*) is through potential fields. In this approach, a real-valued function is computed, the gradient of which is used as the control law for the robot. The biggest drawback with potential field methods is that it is very difficult in practice to compute potential functions that will guarantee global convergence. Theoretical methods exist [85] that make such guarantees, but are difficult to implement and apply to real world environments.

Our approach, on the other hand, builds on top of classical cell decomposition methods used for complete motion planning. Once the cells are determined, we provide a simple and efficient way to place smooth control laws over the decomposition. Thus, exact, smooth feedback motion plans seem to come for free, given the prepaid cost of classical path planning. Over the cell decomposition, it is easy to define local controllers that prevent obstacle collision and guide the system toward the goal state. Our methods place simple local vector fields over individual cells and smoothly blend them together to form a global vector field that has the desired properties of global convergence and obstacle avoidance. By working directly with vector fields rather than computing them from the gradient of a potential function, our methods are able to have a high degree of flexibility from a practical design perspective. In related work, we have also shown that our

methods extend to simple nonholonomic systems [66], [70], which is quite difficult for potential field approaches. Also, the system trajectories produced by our control law are C^∞ smooth. Although this high degree of smoothness is not always necessary, it is often advantageous. Smooth feedback provides a greater level of robustness than non-smooth feedback; it is also more reliable to implement on a physical system. Smooth trajectories also tend to be visually appealing, which is important for applications such as computer animation.

In this paper, we present algorithms for computing feedback laws over two types of cell decompositions. First, we will address bounded, finite-dimensional state spaces with piecewise linear (PL) boundaries; these can be decomposed into convex polytopes. Although our methods work easily in high-dimensional spaces, they are especially practical and fast for two-dimensional problems, which are common in robotics applications. In this case, our algorithms efficiently compute feedback laws in complex polygonal environments. We also present algorithms to construct vector fields over cylindrical algebraic decompositions. From the early years of motion planning research, it has been known that these decompositions can be used to exactly solve the generalized piano movers' problem, which is the one of the most general formulations of the motion planning problem [91]. This is possible because the cells of a cylindrical algebraic decomposition exactly partition the configuration space into regions corresponding to free space or to obstacles. By working with local vector fields over cylindrical algebraic cells, we show how to compute global feedback laws, instead of open loop paths; this is a very basic result, and the most general result on feedback motion planning to date, to our knowledge.

II. Problem Formulation

This section gives the general problem formulation, which follows closely with standard motion planning and control theory notation (see [24], [56], [58] for more details). Denote the *world* as $\mathcal{W} = \mathbb{R}^2$ or $\mathcal{W} = \mathbb{R}^3$. The world contains a given *obstacle region*, which is a closed semi-algebraic set $\mathcal{O} \subset \mathcal{W}$. The *robot* is a 2D or 3D closed semi-algebraic set that may be composed of one or more bodies. The configuration space $\mathcal{C}$ is the smooth manifold corresponding to the space of transformations of the robot. Each $q \in \mathcal{C}$ embeds the robot in $\mathcal{W}$, yielding a subset $\mathcal{A}(q) \subset \mathcal{W}$, which is the closed set of points occupied by the robot. Avoiding collision between $\mathcal{A}(q)$ and $\mathcal{O}$, a *configuration space obstacle region* $\mathcal{C}_{obst} \subseteq \mathcal{C}$ is obtained. The open *free configuration space* $\mathcal{C}_{free}$ is defined as $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obst}$. The classical path planning problem involves computing a path $\tau : [0, 1] \rightarrow \mathcal{C}_{free}$, which connects initial q_i and goal q_g configurations: $\tau(0) = q_i$ and $\tau(1) = q_g$.

Our task differs in two ways: 1) we will leave q_i unspecified, and instead of computing a single path, we compute a vector field that upon integration encodes a family of paths that arrive at q_g from *any* reachable q_i; 2) we replace $\mathcal{C}$ by the more general *state space* X used in control theory. In many cases, it can be assumed that $X = \mathcal{C}$; however, we allow the general case in which coordinates in X correspond to q and its time derivatives $\dot{q}$ (or even higher order derivatives, if necessary). See Chapter 13 of [58]. It is assumed X is a differentiable manifold, and we define the semi-algebraic, closed *state obstacle region* X_{obst} and open *free state space* $X_{free} = X \setminus X_{obst}$, which are derived from $\mathcal{C}_{obst}$ and any additional constraints on time derivatives of q.

Suppose that a *cell decomposition* is given over some connected open region $X_{cd} \subseteq X_{free}$ (for a *complete* method, $X_{cd} = X_{free}$). The cell decomposition is a partition of X_{cd} into "nicely behaved" regions which can be computed using many existing algorithms. The choice depends on the dimension of X and the particular models used for $\mathcal{O}$, the robot, and other constraints on X.

Let x_g be a desired goal point in X_{cd}. Our task is to compute a *smooth feedback plan*, which is a vector field defined over X_{cd} for which all integral curves (trajectories) are C^∞-smooth and converge asymptotically to x_g. Note that even though all integral curves are smooth, V itself may only be smooth *almost everywhere*, which is required because smooth vector fields that converge to x_g do not even exist when X_{cd} is multiply connected.

How can the smooth feedback plan V be used in practice? Suppose that a time-invariant *control system* is expressed over X in the usual manner, $\dot{x} = f(x, u)$, in which u is an *input* at time t, taken from a predefined *input space*, U. Suppose that for every $x \in X_{cd}$, there exists a $u \in U$ such that $V(x) = f(x, u)$. In this case, if it is feasible to solve $V(x) = f(x, u)$ for u at every step, then V immediately specifies what controls to apply to converge to x_g. The most common case in which u always exists is for a *fully actuated* control system with unbounded inputs. Such systems can follow any smooth trajectory arising from V.

If V is designed in a way that cannot be followed by a control system $\dot{x} = f(x, u)$, then it may nevertheless be useful as a guide for *acceleration-based control*, which attempts to track V as closely as possible using the difference between $V(x)$ and $f(x, u)$ as an error term [28], [86]. Rather than tracking a path, as in the use of classical path planning results, the idea is generalized to tracking the vector field, which offers additional flexibility. If such control is not possible, or if it is strictly required that there exists $u \in U$ such that $V(x) = f(x, u)$ at all times, then careful consideration must be given to the particular vectors chosen in V. For example, special fields need to be designed in the case of nonholonomic systems, which we have done in [66], [70]. Also, if a robot cannot follow V due to bounded inputs, it might nevertheless be able to move along the same path in C by simply slowing down.

III. Related Work

In this section, we describe related work in motion planning and control. We begin by outlining open loop motion planning, and continue by describing closed loop methods such as potential field techniques. Finally, we will describe work based on decomposing the environment into discrete cells and creating controllers for each cell, which is our approach in this paper.

A. Open Loop Planning

The development of algorithms that compute open loop trajectories is motivated by the difficulty of finding feedback plans in complex environments. The non-convex constraints induced by obstacles in the environment pose significant problems for classical feedback control techniques [18], [23], [48], [89]. Due to the difficulty of finding closed loop feedback controllers in complex high dimensional spaces, motion planning algorithms attempt to compute only a collision free open loop trajectory; even so, motion planning is PSPACE-hard [83]. Such algorithms have been extensively studied [24], [56], [58]. Most algorithms ignore differential constraints completely, assuming that the robot is a fully-actuated kinematic system (called *free-flying* or *holonomic*). These algorithms include classical motion planning algorithms and many sampling-based algorithms, including the Randomized Path Planner (RPP) [4] and Probabilistic Roadmaps (PRMs) [47]. If the robot is not actually kinematic and holonomic, then the paths produced by these algorithms need post-processing to be transformed into feasible trajectories for the system; this is generally referred to as *decoupled trajectory planning*. Post-processing methods include time-scaling [10], [97], steering [55], [79], or other transformations [25], [35], [57], [93]. In contrast to decoupled trajectory planning, some sampling-based motion planning algorithms directly generate feasible trajectories. Algorithms of this type include Rapidly-exploring Random Trees (RRTs) [60], [61], Expansive Space Trees (ESTs) [45], [46], and PDST-Explore [54]. Other approaches include the use of mixed integer programming, which computes optimal paths for problems with polygonal obstacle constraints and piecewise-affine system dynamics [8], [84], [90]. Both direct and decoupled planning algorithms return open loop trajectories rather than closed loop plans.

One way to improve robustness for open loop paths is to use them as feedforward components in a feedback controller. This has several disadvantages, however. First, paths generated by motion planning algorithms often are of poor quality, having unnecessary sharp turns. This may result in them being difficult to track for a dynamical system. Second, this approach still does not produce a *global* feedback plan; only a local feedback plan in a neighborhood of the nominal trajectory is computed. As a result, it may be difficult to maintain collision avoidance guarantees. Another approach 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 problematic as well. First, it has a very high computational cost, even given the power of modern computers, and may not be suitable for real-time applications. Second, asymptotic convergence to the goal state cannot be guaranteed, even though one might informally expect convergence to occur.

B. Closed Loop Methods

The most common approach to obtaining feedback in the presence of obstacles is to use a potential field. Assume we have a system with $\dot{x} = u$. If a potential field P can be defined that is uniformly maximal on the obstacle boundaries, minimal at the goal state, and whose gradient is non-zero except at the goal state, then setting $u = -\nabla P$ yields convergence to the goal. Simple analysis shows that the potential field P is a suitable Lyapunov function.

The use of potential fields for robot navigation became popular in the 1980s [49], [53]. Khatib's foundational work utilized a potential field over the operational space (workspace) to guide a manipulator or mobile robot to the goal. The basic potential field approach combines a term that is attractive to the goal state with terms that are repulsive with respect to the obstacles. Theory and experiments with different potential fields are given in [110]. Many additional references for potential fields for robot navigation can be found in [43], [74], [117]. The problem with these potential field methods is that they typically have local minima other than the goal state. Any initial condition in the region of attraction of these local minima will fail to reach the goal state. Our algorithms, in contrast, have global convergence guarantees.

Although it is not simple to find potential functions that are free of spurious local minima, it is sometimes possible. Harmonic functions (potential functions which are solutions to Laplace's equation) are guaranteed to be free of such local minima, and can be used for global robot navigation. Connolly *et al.* develop numerical solutions of Laplace's equations for path planning [29]–[32]. For low-dimensional environments, it is possible to discretize the space and consider each node as part of a resistive grid with obstacle boundaries as sources and the goal point as a sink [100], [104], [108]. Wang and Chirikjian simulate steady state heat transfer in [111]. Waydo and Murray use stream functions for navigation in two-dimensional environments [112].

One of the most influential potential field techniques is that of Rimon and Koditschek [85]. They define *navigation functions*, which are potential functions satisfying several technical conditions, and which are guaranteed to be free of spurious local minima. They show how to construct navigation functions for several types of environments, which they call sphere worlds, star worlds, and forests of stars. Following the gradient of the navigation function is guaranteed to lead to the goal state from almost every initial condition (that is, every initial condition except for a set of measure zero). Theoretically, navigation functions can be constructed for a large family of configuration spaces, although this can be very difficult to implement in practice. Navigation functions have been extended to the case of multiple, nonholonomic robots [71], [72], [102], [103].

There are a number of other local navigation approaches based on potential fields. These include the Virtual Force Field (VFF) [12] and the Vector Field Histogram (VFH) [13] and their extensions, VFH$^+$ [107] and VFH* [106]. These methods build a potential field online, using range sensor measurements. This online potential field can be used to avoid obstacles and move toward the goal, but convergence is not guaranteed. The Curvature Velocity Method [52], [98] and the Dynamic Window Approach [36] choose controls at each time step that are optimal over the set of admissible controls

(i.e., those for which the robot can always halt without hitting an obstacle). The optimality criteria can be chosen to cause the robot to travel towards the goal, and the robots generally move quite rapidly while observing the safety constraints; once again, however, convergence is not guaranteed. The dynamic window approach is extended in [17], [82], [99].

Potential field methods have also been integrated with sampling-based motion planning algorithms in a variety of ways. The sampling-based neighborhood graph (SNG) covers the free space with balls, each of which is equipped with a local navigation function that is guaranteed to convey the robot into a ball nearer to the goal state [115]. Bohlin used Green kernels to compose a workspace potential using samples from $SE(3)$ [11]. Elastic roadmaps build dynamic roadmaps using features in the workspace [116]. Connectivity between roadmap milestones is determined by local potential functions. Elastic roadmaps have been successfully applied to challenging problems in dynamic environments, but they lack completeness guarantees. Approaches like these can be viewed as hybrid control systems, which are discussed more fully below.

Velocity field control is an alternative to potential field methods. Velocity field control places a vector field over the state space directly, rather than computing it as the gradient of a potential field. One motivation for this approach is that it allows task specification (e.g., trajectory or contour following) without time parameterization. It was introduced by Li and Horowitz [62]–[64]; stability of the system is demonstrated using notions of passivity. Velocity field control has been applied frequently to robot manipulators [21], [78], with the velocity field specified over the operational space of the manipulator. Velocity fields have also been applied to wheeled mobile robots [33], [113], [114]. Although stability and convergence results are obtained for systems with nontrivial dynamics, velocity field methods do not consider environments with obstacles.

Another approach is to use numerical techniques to compute an approximate optimal value function on the space, which then serves as a potential field. In this case, not only is feedback achieved, but also approximately optimal trajectories [9], [59], [77], [94], [95], [105]. The time and space complexity of these algorithms are exponential in the dimension of the state space, for fixed sampling or discretization resolution; therefore, the curse of dimensionality prevents the application of this approach beyond a few dimensions.

As we have seen, the basic problem with these methods is that they generally either do not have formal convergence and obstacle avoidance guarantees, or they are not simple to implement and use for robots operating in complex real-world environments. Our goal is to do better than this: to construct feedback laws which have strong convergence and safety properties and which are also highly efficient and practical.

C. Hybrid Control Systems and Sequential Composition

A hybrid control system is one that incorporates both discrete and continuous dynamics. The control system operates in one of a distinct number of modes, and switches or jumps between them when certain conditions are met. Formal models of hybrid systems have been defined and studied [1], [14]–[16], [65], [109]. One particular type of hybrid controller is based on sequential composition of funnels [19], [73], [86]. In this framework, a collection of controllers is developed, each of which converges to a goal set that 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.

In the case where the environments are polygonal (a common scenario), one approach is to divide the environment into convex cells and use local controllers on each cell. If the controller for each cell funnels the robot to an appropriate edge of the cell, then the controller for the next cell can take over. When the goal cell is reached, the local controller causes the robot to converge to the goal state. The case of piecewise affine hybrid systems has been studied extensively, considering control on simplices [7], [38], [39], [87], rectangles [6], [51], or general polytopes [40], [41] (see also references in these works). Since affine functions over simplices are exactly determined by their value at the vertices of the simplices, it is possible to prove reachability and controllability results simply by solving linear inequalities. Fainekos *et al.* show how to use controllers such as these in an integrated approach capable of satisfying complex linear temporal logic specifications [34].

Conner *et al.* use local potential fields to define control policies on individual polygonal cells [28]. To define the field, they use 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 (i.e., the gradient of the potential field, and thus the control policy, is continuous). Putting together the individual "component control policies" 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 second order systems. They also use the composition of funnels technique to deploy control policies for convex-bodied robots with nonholonomic constraints [27].

Finally, our work can also be viewed as the sequential composition of funnels, in which the environment is decomposed into appropriate cells and local control policies defined over each cell [67]–[69]. Our methods give stronger smoothness results than the above methods, and have extended them to a unicycle robot [66] and a car-like robot [70].

IV. Smooth Feedback on Convex Cell Decompositions

In this section, we describe how to construct a smooth feedback plan on a d-dimensional cell complex embedded in $\mathbb{R}^d$, in which each cell is an open convex polytope. An earlier version of this work appeared in [67]. As we have already discussed, this might result from a decomposition of a d-dimensional space with a piecewise linear boundary. If the space is described using an arrangement of hyperplanes, an acceptable decomposition is simply to use the complement of the arrangement. An alternative decomposition with potentially

fewer cells is vertical decomposition [42], [92]. The input to our algorithm is the cell complex and a goal state x_g. As discussed in Section II, the task is to construct a vector field on the cell complex such that the integral curves are smooth, avoid obstacles, and converge to the goal state.

A. Description

To compute the desired smooth feedback plan, our algorithm performs the following steps:

1) Given the cell decomposition, compute a discrete plan over the cells.
2) Design local controllers (vector fields) that avoid obstacles and are consistent with the discrete plan.
3) Smoothly combine the local controllers to obtain a global controller that has the desired properties.

We discuss these each in turn.

Discrete plan computation: Let a d-dimensional cell in a cell complex be called a *d-cell*. Suppose that a connected cell complex is given in which a collection of d convex cells are specified along with $(d-1)$ convex cells at the boundaries between pairs of adjacent d-cells. Under the convention that X_{cd} is open, these cells are sufficient for motion planning because all paths between two d-cells go through a shared $(d-1)$-cell. Since we do not need to consider lower dimensional cells, we will henceforth use the term *cell* to refer to a d-cell and *face* to refer to a $(d-1)$-cell. Define the connectivity graph to be the graph that has a vertex for each cell (d-cell) and an edge between two vertices if and only if the corresponding cells share a face ($(d-1)$-cell) on their boundaries. Compute a discrete plan over this graph such that following the plan from any vertex leads to vertex corresponding to the cell containing the goal state. A variety of graph search algorithms can be used for this purpose, with or without optimality criteria. For example, breadth-first search can be used, with a corresponding linear bound in execution time. Alternatively, edge weights can be assigned using distance between cell centroids, and Dijkstra's algorithm or dynamic programming can be used to find cell paths that induce shorter paths through the environment. The resulting directed graph defines a *successor* for every cell except the goal cell. The successor of a cell is the next cell on the path to the goal cell; the shared face is called the *exit face* of the first cell. Each cell with a successor is termed an *intermediate* cell, in distinction with the goal cell, which has no successor. See Figure 1 for an illustration.

Local vector fields: The directed graph and corresponding successor relations define a high-level discrete plan. Now, we define local vector fields that are consistent with this plan. To do so, we define two types of vector fields: those corresponding to cells in the decomposition, which we call *cell vector fields*; and those corresponding to faces, which we call *face vector fields*. Intuitively, the purpose of a cell vector field is to guide the robot through the cell to the exit face, which leads to the successor cell. The purpose of the face vector fields is to guarantee avoidance of the X_{cd} boundary and to guarantee adherence to the discrete plan; in other words, the face vector fields prevent the robot from crossing a cell face corresponding to the X_{cd} boundary or an improper cell transition, instead causing the robot to cross the exit face into the successor cell. For the sake of clarity, we will delay discussing the formal requirements for these vector fields. For now, consider the face vector fields to be normal to their corresponding faces and oriented in the appropriate directions, and the cell vector fields to always point toward the exit face. In the case of the goal cell, all face vector fields point inward and the cell vector field always points at the goal state. See Figure 2 for an illustration of face vector fields.

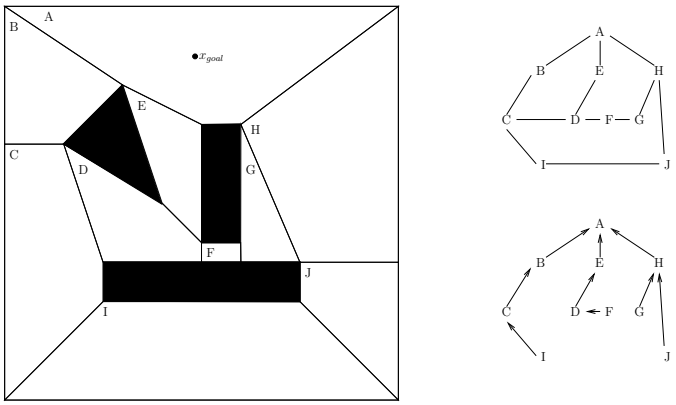

Fig. 1. An environment decomposed into convex cells, and the corresponding connectivity graph and discrete plan.

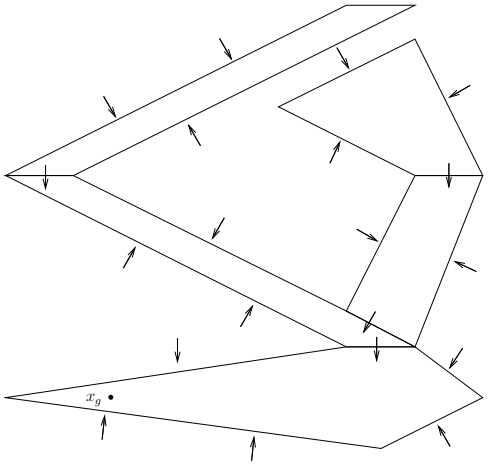

Fig. 2. Vector fields assigned to the faces.

Smooth interpolation: Now we proceed to the third task of our algorithm, which is to interpolate between these local vector fields to obtain a global vector field that has the desired smoothness and convergence properties. Consider a single cell; we then have a single cell vector field V_c and a set of face vector fields $\{V_{f_i}\}$. We will form a vector field V by interpolating between these vector fields; we will do this in such a way that V equals the face vector fields on their corresponding faces. This guarantees that the vector field will be continuous across cell boundaries (we will see later that all derivatives will match across cell boundaries as well, yielding smoothness). Interpolation is greatly simplified if it is only pairwise, rather than interpolating between all of the local

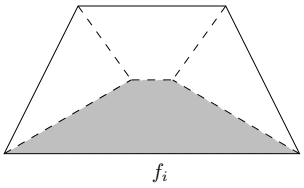

Fig. 3. A cell, partitioned using the generalized Voronoi diagram. The dashed lines are the GVD surface, and the shaded region is the region of influence (Voronoi region) of face f_i.

vector fields simultaneously. A natural choice for this is to use the generalized Voronoi diagram (GVD) of the cell [81], [88]. The GVD is formed by partitioning the cell into *Voronoi regions*, of which there are one per face. The Voronoi region of each face is defined to be the set of points inside the polytope that are closer to that face than to any other face; we refer to the Voronoi region of a face as its *region of influence*. The *GVD surface* is the set of points which are equidistant from two or more faces of the cell. Since the faces are $(d-1)$-dimensional hyperplanes, the GVD surface is the union of subsets of hyperplanes, each of which is equidistant from a pair of faces. See Figure 3.

For any point in the region of influence of face f_i, V will be an interpolation of V_{f_i} and V_c. On the face itself, $V = V_{f_i}$; the rest of the boundary of the region of influence is contained in the GVD surface, and we assign $V = V_c$. In order to smoothly interpolate over an individual region, we need a smooth function which is uniformly zero on the face f_i and uniformly one on the GVD surface. This function should be smooth, except on the $(d-2)$-dimensional intersection of f_i and the GVD surface. Lack of smoothness at these points will not adversely affect our method, because we have already indicated that all system trajectories move from cell to cell through $(d-1)$-dimensional faces, not through $(d-2)$-dimensional ones. We construct smooth interpolating functions using *bump functions*, a common construction in differential geometry, which 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 $\phi : X \to [0,1]$ such that:*

1) *ϕ has support contained in U.*
2) *$\phi(x) = 1$ for every $x \in K$.*

Numerous bump functions are known. We will introduce one that transitions smoothly from 0 to 1 on the unit interval and is simple to express. First, define

$$\lambda(s) = (1/s)e^{-1/s}. \tag{1}$$

The bump function is then defined as

$$b(s) = \begin{cases} 0 & s \leq 0 \\ \frac{\lambda(s)}{\lambda(s)+\lambda(1-s)} & 0 < s < 1 \\ 1 & 1 \leq s \end{cases} \tag{2}$$

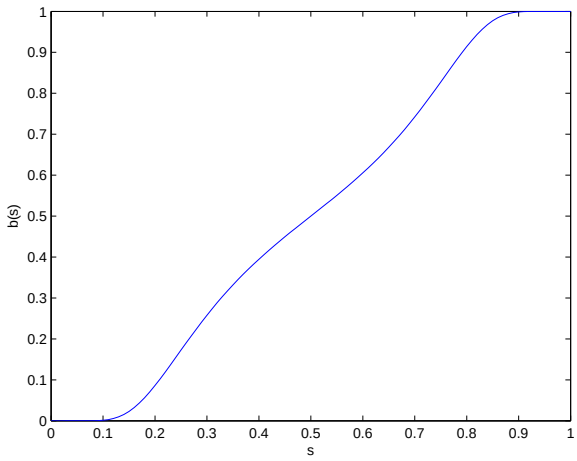

Fig. 4. A plot of the bump function given in (1), which is used to smoothly interpolate between vector fields.

An illustration of this bump function is given in Figure 4. The bump function has the important property that all derivatives equal zero at the endpoints of the unit interval.

Proposition 1 *For any i, $d^ib/ds^i(0) = d^ib/ds^i(1) = 0$.*

Proof: By inspection of (2), it is clear that $d^ib/ds^i(0^-) = d^ib/ds^i(1^+) = 0$, since b is identically 0 for $s < 0$ and identically 1 for $s > 1$. Therefore, consider the derivatives in the open unit interval. From Equation 1, straightforward application of L'Hôpital's rule yields

$$\lim_{s\to 0^+} \frac{d\lambda(s)}{ds} = \lim_{s\to 1^-} \frac{d}{ds}(\lambda(1-s)) = 0.$$

All higher derivatives of λ are likewise zero, by successive applications of L'Hôpital's rule. Equation 2 yields

$$\begin{aligned} \frac{db(s)}{ds} &= \frac{\frac{d\lambda}{ds}\left(\lambda(s)+\lambda(1-s)\right) - \lambda(s)\left(\frac{d\lambda}{ds} + \frac{d}{ds}\lambda(1-s)\right)}{(\lambda(s)+\lambda(1-s))^2} \\ &= \frac{\frac{d\lambda}{ds}\lambda(1-s) - \lambda(s)\frac{d}{ds}\lambda(1-s)}{(\lambda(s)+\lambda(1-s))^2}. \end{aligned}$$

As $s \to 0^+$, both $\lambda(s)$ and $\frac{d\lambda}{ds}$ go to zero; hence, $\frac{db}{ds} \to 0$. Similarly, as $s \to 1^-$, both $\lambda(1-s)$ and $\frac{d}{ds}\lambda(1-s)$ go to zero; again, $\frac{db}{ds} \to 0$. All higher derivatives go to zero in the same way. ■

The parameter s we use for the bump function is the product of a number of analytic switches, which is smooth over the interpolation region (the region of influence of the face). For any point p in the Voronoi region of face f_i, let

$$s(p) = 1 - \prod_{j\neq i} \frac{\rho(p,f_j) - \rho(p,f_i)}{\rho(p,f_j)}, \tag{3}$$

in which ρ is the Euclidean distance metric in $\Re^n$. This function is smooth (except on the $(n-2)$-dimensional boundary of the cell), and has the desired property of being identically equal to zero on the cell face. The rest of the boundary of the region of influence is the GVD surface, on which the equation $\rho(p,f_j) = \rho(p,f_i)$ is satisfied for some j. Therefore, (3) is

identically equal to one on this boundary. Note that explicit computation of the GVD is *not* required for this construction; it is simple to determine which face's Voronoi region any particular point is in, by computing the distance to each face of the cell. The point is in the Voronoi region (region of influence) of the face which is closest to it.

Putting the pieces together, the overall vector field V is defined as:

$$V(p) = \text{unit}\left(b(p)V_f(p) + (1 - b(p))V_c(p)\right), \qquad (4)$$

in which V_f is the face vector field for that point, V_c is the cell vector field, b is the bump function with $b(p)$ shorthand for $b(s(p))$, and unit is a normalization function, ensuring that V is a unit vector field.

The approach needs only slight modifications for the goal cell. In this case, the GVD is not used to partition the cell; instead, the region of influence of a face is defined to be the (interior of) the convex hull of the face together with the goal point. This clearly results in a subdivision of the cell. The interpolating function, then, goes from zero on the face to one on the rest of the boundary of the cell. Since the boundary consists of a number of hyperplanes (as in the previous case), it is easy to compute the necessary distances and the value of the interpolating bump function.

By showing how to construct the vector field on both the goal cell and intermediate cells, we have constructed a vector field over the entire cell decomposition. Next, we will formally specify sufficient conditions for the face and cell vector fields and prove that the resulting vector field V satisfies the requirements of a smooth feedback plan.

B. Theoretical Results

We begin by formally defining cell vector fields for intermediate cells. Note that the GVD surface is the union of a set of faces, each of which is a subset of a $(d-1)$-dimensional hyperplane equidistant between two cell faces.

Definition 2 *Let C be a convex cell with exit face f_x, and consider the GVD of C. A* cell vector field *V_c is a smooth unit vector field on C that satisfies the following:*

1) *For each point $x \in C$, there exists a $y \in f_x$ and $\alpha \in \mathbb{R}$ such that $V_c(x) = \alpha(y - x)$.*
2) *Let h be a GVD face, with normal n. If $V_c(x) \cdot n = 0$ for some $x \in h$, then $V_c(x) \cdot n = 0$ for all $x \in h$.*
3) *The directed transition graph induced by this choice of vector fields is acyclic and every path through this graph terminates at the node corresponding to the exit edge.*

Although this definition permits many different types of cell vector fields, we will consider a more narrow class of cell vector fields in practice. Consider a convex cell C with exit face f_x with outward pointing normal n_x, and let $\bar{C}$ be the (possibly unbounded) cell resulting from the removal of f_x from C. Let $V_c(x) = \text{unit}(p - x)$, in which $p \in \bar{C} \setminus C$ is fixed. A variety of similar constructions are possible. See Figure 5 for an illustration.

Proposition 2 *As defined above, V_c is a cell vector field.*

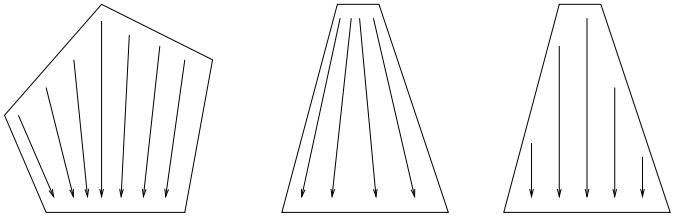

Fig. 5. Three possible cell vector fields. Each points toward the exit face at the bottom of the cell.

Proof: Each integral curve of V_c is a straight line; it is simple to see that each integral curve crosses the exit face, f_x; hence, the first condition is satisfied. For the second condition, note that each face of the GVD is a portion of the hyperplane separating two faces of C. If V_c is constant, then the second condition is clearly satisfied. If V_c is otherwise, then for some GVD face with normal n, the sign of $V_c \cdot n$ is fixed for any x on the GVD face; this can be easily verified from the definition of V_c above. For the third property, we have already stated that each integral curve is a straight line that crosses f_x. Together with the second property, this directly implies the third property. ■

The definition for a face vector field is simple. Again, let C be an intermediate cell with exit face f_x. For any face f, let the associated normal vector n be inward pointing if $f \neq f_x$, and outward pointing for $f = f_x$. For any face $f \neq f_x$, denote the hyperplane of points equidistant to f and f_x (the bisecting hyperplane) by b_f. Denote the unit normal vector of this hyperplane to be n_{bf}, and let it be oriented so that $n_{bf} \cdot n_x > 0$, for n_x the normal of the exit face.

Definition 3 *A* face vector field *corresponding to a face f is a smooth unit vector field V_f such that for every $p \in f$, $V_f(p) \cdot n > 0$. If $f \neq f_x$, the condition $V_f(p) \cdot n_{bf} > 0$ must hold for every p in the closure of the region of influence of f; if $f = f_x$, $V_f(p) \cdot n > 0$ must hold.*

We now show that these broad conditions on the cell and face vector fields are sufficient for the integral curves of V to reach the exit face of the cell in finite time.

Theorem 1 *Under Definitions 2 and 3 above, all integral curves of V reach the exit face in finite time.*

Proof: Take any point p in a region of influence of some $f \neq f_x$. It is clear from the definition of V_c that there exists some ϵ_1 such that $V_c(p) \cdot n_{bf} \geq \epsilon_1$. With respect to the face vector field, we know that $V_f \cdot n_{bf}$ is bounded away from zero on a closed set, which implies that there exists some ϵ_2 such that $V_c(p) \cdot n_{bf} \geq \epsilon_2$. Therefore, the overall vector field V will satisfy $V(p) \cdot n_{bf} \geq \epsilon$ for $\epsilon = \min(\epsilon_1, \epsilon_2)$, everywhere on that region of influence. This implies that the integral curve containing p will reach the bisecting hyperplane b_f in finite time, unless it first reaches a GVD face. If it reaches a GVD face, then it crosses into the region of influence of another cell, and will never return to the first region of influence, by property (3) of Definition 2. Applying this property repeatedly,

we see that the integral curve will reach the region of influence of the exit face in finite time.

Assume that the integral curve has reached the region of influence of the exit face. The integral curve then either reaches the exit face or some GVD face in finite time, because the distance to the exit face is decreasing at a rate that has a positive lower bound. The integral curve cannot reach another GVD face by the third condition of Definition 2; therefore, it reaches the exit face in finite time after entering the region of influence of the exit face. Therefore, all integral curves in the cell reach the exit face in finite time. ■

We have shown that for every intermediate (non-goal) cell, every integral curve of the generated vector field will reach the exit face of that cell, and hence continue to the next cell. Consequently, we have shown that all integral curves will reach the goal cell. However, it remains to be shown that all integral curves in the goal cell will reach the goal point. The argument is much the same.

In the goal cell, we use a different definition of the cell vector field. Formally, we require that for any point $p \neq x_g$ in the goal cell, $V_c(p) \cdot (x_g - p) > 0$; also, we require $V_c(x_g) = 0$. Practically, we use a cell vector field that is always oriented toward x_g,

$$V_c(p) = b(||x_g - p||)\text{unit}(x_g - p), \tag{5}$$

to satisfy this condition. This is smooth, satisfies the inner product requirement, and decays to zero at the goal point.

Theorem 2 *All integral curves in the goal cell C_g asymptotically converge to the goal point.*

Proof: This statement is proven similarly to Theorem 1. Both the face and cell vector fields satisfy an inner product constraint guaranteeing that at any point $p \neq x_g$, $V(p) \cdot (x_g - p) \geq \epsilon$ for some $\epsilon > 0$. The only place where $V(p) \cdot (x_g - p) = 0$ is at the goal point x_g; therefore, every integral curve asymptotically converges to x_g. ■

Given the previous theorems, the following theorem holds true:

Theorem 3 *The integral curves of the vector field V, defined over $X_{cd} \subseteq X_{free}$, asymptotically converge to the goal state x_g.*

Proof: From Theorem 1, any integral curve in an intermediate cell proceeds to the exit edge and thus continues to the successor cell. All integral curves consequently proceed to the goal cell in finite time. Theorem 2 then implies that the integral curves asymptotically converge to the goal state. ■

We emphasize that the conditions we have given on the face and cell vector fields are not necessary, but sufficient. Other than those we have outlined, there are many combinations of face and cell vector fields that will yield the same result. If a choice of vector fields is made that does not satisfy these sufficient conditions, it may still be possible to show that convergence follows. Arguments like those made above would likely be sufficient to verify convergence: ensuring that the combination of face and cell vector fields will always make "sufficient progress" (e.g., satisfy an inner product constraint) in each intermediate cell, and will converge to the goal state once the goal cell is reached.

Having fully described the construction of V and shown that the integral curves converge to the goal state, we now prove the following:

Theorem 4 *All integral curves of V are smooth.*

Proof: As we have defined them, all local face and cell vector fields are smooth. We have already argued that the bump function $b(s)$ is smooth. The parameter function s is smooth on every Voronoi region, except on a set of measure zero (the $(d-2)$-dimensional boundary of the cell face). Integral curves never pass through these points, because every integral curve in a particular cell passes to the successor cell through the open $(d-1)$-dimensional face between them. The fact that all derivatives of the bump function equal zero for $s = 0$ and $s = 1$ guarantees that the vector field (and, correspondingly, its integral curves) are smooth across cell boundaries and across the GVD surface within each cell. Therefore, the integral curves of V are smooth. ■

C. Efficiency

We have claimed that our method is extremely fast to compute. There are two primary computational costs. First, there is the cost to compute the component vector fields given an environment and a goal state; this is the *precomputation* cost. Second, there is the problem of computing the value of the vector field at a given point; this is the *execution* cost. These can both be done quickly. We will give the asymptotic complexity of these algorithms, but we emphasize that the constants in the asymptotic analysis are quite small; these methods are very efficient in practice as well as in theory.

First, consider the precomputation phase. If breadth-first search is done on the graph corresponding to the cell complex, the successor of each cell can be found in $O(n)$ time, in which n is the number of d-dimensional cells in the decomposition. If a Dijkstra-like approach is used, the complexity becomes $O(n \log n)$. The face vector fields can be assigned in linear time if perpendicular face vector fields are used. The cell vector fields likewise require only linear time, since they can be assigned to point to the centroids of the exit faces. Hence determining the component vector fields, given a cell complex and its connectivity graph, can be done in linear time.

Second, consider the execution cost. If the cell in which the query point lies is unknown, then a point location query must be performed to determine in which cell the point lies. This can clearly be done in linear time, and may be answered in logarithmic time if some preprocessing of the cell decomposition is done. In two dimensions, the optimal preprocessing bound is $O(n)$ time, but practical algorithms typically require $O(n \log n)$. Also, only linear space is required in two dimensions. A good algorithm for this purpose is Kirkpatrick's triangulation refinement method [50]. In higher dimensions, the results are not as good: logarithmic query time (more precisely, $O(d \log n)$, in which d is the dimension) can

be attained, but only at the cost of exponential space: $O(n^{2^d})$ [37].

If the cell of the query point is known, it requires linear time (in the number of faces of the cell) to compute the vector field value, because computing the bump function parameter requires computing the distance to each face of the cell. In practice, the number of faces of any cell is so small that the cost of computing the vector field value is practically negligible. If there is no error in following the vector field, only a single point location query must be performed to compute an entire trajectory. Consider two successive query points: they must either lie in the same cell, or the second one lies in the cell that is the successor to the first one. This is guaranteed to hold as long as we assume that the vector field is queried at a high enough rate, which is a weak assumption. The most reasonable assumption is that the vector field is queried almost continuously (as in real time control), which will result in the condition holding true. In the presence of error, this may not always be the case; however, we expect that it should typically hold in practice, assuming that the error is small.

We may also assume that the cell complex is not given to us directly, but that it must be computed by decomposing a general polygonal environment. We can do this for any dimension using vertical decomposition (an arbitrary-dimension version of trapezoidal decomposition); see [42] for details. If we restrict ourselves to the two-dimensional case, there are many ways to decompose polygon into convex pieces. One option is Keil's algorithm for computing a convex decomposition with a minimal number of pieces. Keil's algorithm requires $O(nr^2 \log n)$ time, in which n is the number of vertices and r the number of reflex vertices. Triangulation can be done in linear time [22], and a practical implementation based on Seidel's algorithm is available, which requires $O(n \log^* n)$ time [80]. In practice, these algorithms can decompose even large and complicated environments in milliseconds, on modern desktop computers. Shewchuk's Triangle library [96] is extremely useful for this; it produces high-quality Delaunay triangulations from general polygonal environments. The output can be used directly, or post-processed using an algorithm such as Hertel-Mehlhorn [44] to obtain larger and fewer convex cells.

D. Discussion and Computed Examples

Thus far, we have focused almost exclusively on theoretical considerations. While the algorithm and its theoretical properties are the primary focus of this work, we also wish to present several computed examples. We consider these examples to be proof of concept; they have been chosen to illustrate our approach and, we hope, to convey intuition about the feedback controllers computed by it. We do not attempt to make precise statements about performance, or to "push the limits" of what is practically feasible in a hard or soft real time setting. Computed examples are given in Figures 6-8.

We will also briefly discuss some issues associated with using our approach to compute "good" feedback plans for practical mobile robot applications. In the discussion of theory above, we attempted to make the formal conditions for convergence guarantees as broad as possible. The reason for this is

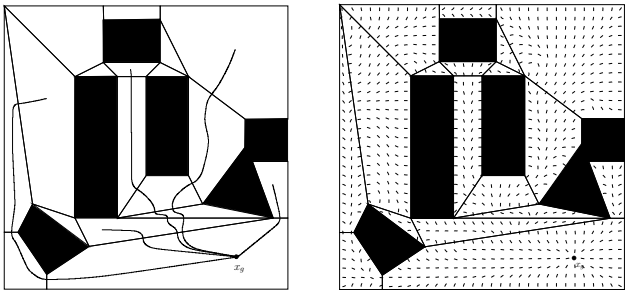

Fig. 6. A computed example; the left figure shows several system trajectories, and the right figure illustrates the entire vector field.

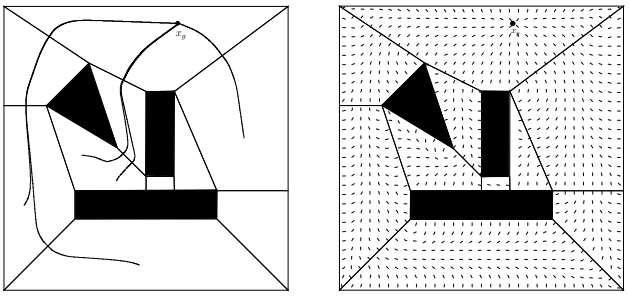

Fig. 7. A second computed example.

that we want to allow as much room as possible for flexibility at the point of implementation, while still maintaining provable guarantees. Choice of cell decomposition and selection of face and cell vector fields are important for our algorithm, but may be driven by application-specific concerns in a particular context. We will outline some issues we have encountered in our preliminary experiments, and comment on reasonable options for other implementations and applications.

Designing face and cell vector fields: In Section IV-B, we stated fairly general conditions on the face and cell vector fields under which convergence is guaranteed. This permits a great deal of application-level design flexibility. To this end, we will outline several approaches for designing face and cell vector fields. We also give several concrete examples to illustrate the impact of the choice of convex decomposition and face and cell vector fields on the "quality" of the resulting paths.

Consider a face vector field for some face other than the exit face of a cell. To satisfy the necessary conditions for convergence, such a vector field must be directed inward at the face itself and must satisfy an inner product constraint with the normal of the face that is equidistant from the face and the exit face. Although these requirements are quite loose, we will consider only the class of constant vector fields, with the goals of simplicity and practical performance. A great deal of design freedom still remains under the constant vector field restriction.

First, consider the case of a constant face vector field with only the restriction that it must be inward-pointing on the face itself. In other words, we have only the constraint $V_f \cdot n_f > 0$, in which n_f is the inward-pointing normal of the face. With this much freedom, there are several obvious ways to choose

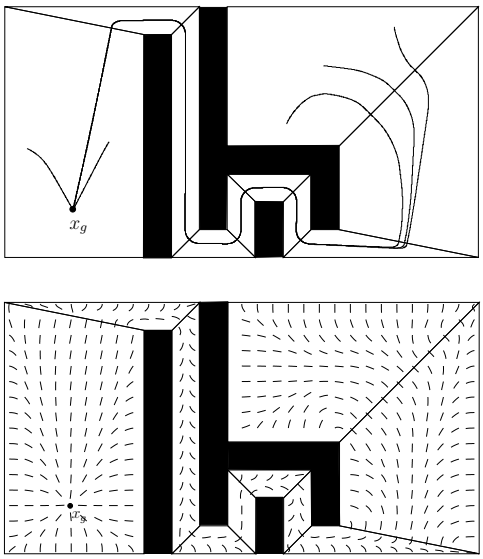

Fig. 8. A third computed example.

V_f that might be advantageous in terms of generating high-quality paths. Different applications might choose any of the following, depending on their specific requirements:

- Choose the vector field to point toward the exit face as much as possible, to promote short paths.
- Choose it to point *as far away from* the exit face as possible, to avoid the sharp turns that the first approach might induce.
- For simplicity, just make each face vector field perpendicular to its face.
- Incorporate additional information. For example, compute the centroid of the cell or of the exit face, and direct the vector field toward it.

In different situations, each of these approaches could offer advantages; this greatly depends on the particular application. The first will tend to induce short paths, but with high curvature. The second takes longer paths, but has fewer sharp turns. Note, however, that *any* fixed vector field will have integral curves with sharp turns; this is an inescapable consequence of the uniqueness of solutions to differential equations (integral curves are solutions to the differential equations corresponding to the vector field). This is a fundamental difference between feedback approaches and open loop motion planning. The third approach is simple enough to be used as a default, "out of the box" method before attempting improvements or optimizations. Its weakness, however, is high sensitivity to the underlying decomposition; this can be seen in the computed examples, for which perpendicular vector fields were used. The fourth approach also tends to increase separation from the X_{cd} boundary, but is less extreme than the second approach. We have not performed sufficient experiments to make definitive comparisons between these methods (and there are many others, to be sure); we leave that for future work.

Convergence can be guaranteed for any of the above approaches. The constraints for guaranteeing convergence are all simple inner product constraints (i.e., each constraint requires

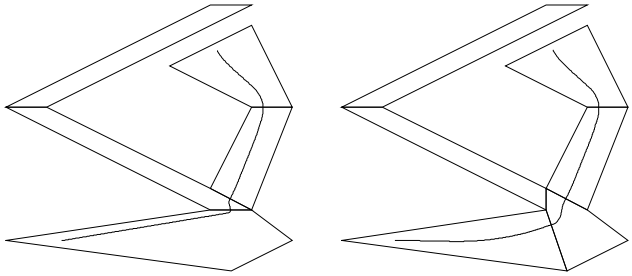

Fig. 9. The influence of different convex decompositions. On the left, a path with sharp turns arising from the choice of decomposition and perpendicular face vector fields; on the right, the artifacts eliminated through a better decomposition.

that the vector field have positive inner product with the normal of some hyperplane). Therefore, it is usually straightforward to take a broad concept such as "point away from the exit face as much as possible" and translate it into an admissible face vector field. If the desired direction of travel violates a dot product constraint, simply project that vector onto each violated constraint. The result is the direction that moves in the desired direction as much as possible while satisfying all constraints necessary to make convergence guarantees.

Cell vector fields can also have a significant impact on path quality. As we discussed above, the cell vector fields can sometimes be constant vector fields; otherwise, we generally choose a point p and let $V_c(x) = \mathrm{unit}(p - x)$. We described above the conditions on the placement of p. The distance from the exit face to p has a significant impact on the resulting paths; the closer p is to the exit face, the more the integral curves tend to bunch together when leaving the exit face. An extreme choice is to place p on the exit face itself; this strongly influences the integral curves to leave the cell near p. Choosing p near the centroid of the exit face can be beneficial. Caution must be exercised, however, because improper placement of p on the exit face can lead to undesirably sharp turns.

Finally, the choice of convex decomposition can greatly affect the quality of the resulting paths, especially near the goal cell. This is particularly important when the face vector fields are chosen to be perpendicular to the edges. See Figure 9 for an illustration of this point.

Dynamic Environments: As we have described it, our algorithm applies to static, known environments. For practical applications, however, this will not necessarily be the case. Environments can change continuously over time, as in the case of moving obstacles, or discretely, as when a door opens or closes in a building environment. We do not wish to explicitly integrate environment uncertainty into our algorithm; we assume, then, that we have complete knowledge of the environment at all times, even though it may change unpredictably. A practical application can easily integrate our approach with sensor-based map updates and higher level exploration behavior; therefore, this assumption is sufficient for our interests.

We will consider two types of changes to the environment. First, take the case of discrete changes in the environment; in our model, this takes the form of new faces being introduced

or removed that change the topology of the environment. If the face under consideration corresponds to a face in the convex decomposition, then the decomposition remains unchanged; the only change is that an open face has become an face on the X_{cd} boundary, or vice versa. Hence, the most our algorithm has to do is update the connectivity graph and search it to obtain a new directed graph defining updated successor relations. If the edge introduced does not correspond to a face in the decomposition, then the decomposition must be recomputed such that this is the case. As before, the connectivity graph must then be searched to generate the successor relations. This possibility indicates that if there is prior knowledge about what faces can be removed or introduced into the environment, this should be incorporated into the initial decomposition.

Second, the environment can change in a more general way; entire obstacles might move, or gross changes to the environment could be made (as in the case of sensor-based map updates). These changes can be either large or small, and may or may not affect the topology of the space. If the changes are local, then it may be possible to repair the decomposition by recomputing the cells in the neighborhood of the change. If the change is large, then the entire decomposition may have to be recomputed. In small environments, it is likely that recomputing the entire decomposition from scratch will be more efficient than attempting to make local repairs; in large environments, this may not be the case. As we have already mentioned, there are efficient algorithms to perform the convex decomposition; in two dimensions, it can be done for many environments in just milliseconds. This indicates that even in dynamic environments, real time performance can be achieved.

V. Smooth Feedback on Cylindrical Algebraic Decompositions

Up to now, we have discussed how to construct smooth feedback plans on cell complexes in which each cell is a convex polytope, and we have shown how our methods can be applied effectively in practice. An earlier version of this work appeared in [68]. In this section, we consider the same problem on a different type of decomposition. We describe the construction of smooth feedback plans on cylindrical algebraic decompositions, which greatly extends the results of the previous section. Since cylindrical algebraic decompositions can be used to solve a very general class of motion planning problems, our algorithm demonstrates how to compute smooth feedback for the same class of problems. In addition, the feedback laws can be computed *efficiently*; precisely, they can be computed in $O(n)$, in which n is the complexity of the decomposition (the number of d-dimensional cells in the decomposition). Since the number of cells in a general cylindrical algebraic decomposition can be doubly-exponential in the dimension of the space, efficient computation of smooth feedback with respect to the decompositions still implies a very pessimistic overall time bound. However, there exist problems that admit cylindrical decompositions that have much better complexity bounds (e.g., planning for the ladder [3] or a polygon translating and rotating in the plane [2]). In cases such as these, our algorithm has potential for practical implementation and use.

A. Cylindrical Algebraic Decomposition

To generate a smooth feedback plan over the entire cell decomposition, our algorithm will make use of the cells' cylindrical structure. Therefore, we will describe cylindrical algebraic decompositions as a preliminary to the presentation of our algorithm.

A *cylindrical algebraic decomposition* (CAD), also known as a *Collins Decomposition* [26], of $\mathbb{R}^n$ is defined in the following inductive way (see [5] for a more formal definition):

Definition 4

1) *A cylindrical algebraic cell C_1 of level one is either an interval (a, b) or a point a.*
2) *A cell C_n of level 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 $f, g \in \mathbb{Q}[x_1, x_{n-1}]$ are polynomials over the field of rational numbers.*

The cells' cylindrical structure is apparent from the definition. For a set of polynomials $\mathcal{P}$ taken from the set $\mathbb{Q}[x_1, \ldots, x_n]$, 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 polynomial in the cardinality of $\mathcal{P}$, as well as in the algebraic degree of the members of $\mathcal{P}$; however, it is doubly exponential in the dimension.

In addition to proposing the decomposition, Collins gave an algorithm to compute it [26]. 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 a projection that preserves the zeros of $\mathcal{P}$ as well as the intersections of the members of $\mathcal{P}$. 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 C_1. In the second phase, the cells of C_1 are lifted into $\mathbb{R}^2$, becoming cylinders that are partitioned based on the critical points of the polynomials that 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 [5], [91], the unbounded cells can be treated as the others by considering the set of polynomials to include $x_i = \pm\infty$, for $i = 1, \ldots, n$. More details can be found in [5], [58], [76]. A (very) simple illustration can be seen in Figure 10. Additionally, the algorithm can compute a single point in each cell of any dimension i, $1 \leq i \leq n$; such points are called *algebraic points* .

Schwartz and Sharir showed how to use the CAD algorithm to solve the generalized piano movers' problem. In this problem, the robot $\mathcal{R}$ and the obstacle region $\mathcal{O}$ are specified as semi-algebraic sets; a collision-free path must be found from an initial configuration to a goal configuration, if one exists. Additionally, there may be more than one robot, and the robots may be connected in a kinematic tree. For this problem, $\mathcal{C}_{free}$ and $\mathcal{C}_{obst}$ are semi-algebraic in the configuration space, and each cell in the cylindrical algebraic decomposition of the configuration space is either completely contained in $\mathcal{C}_{free}$ or

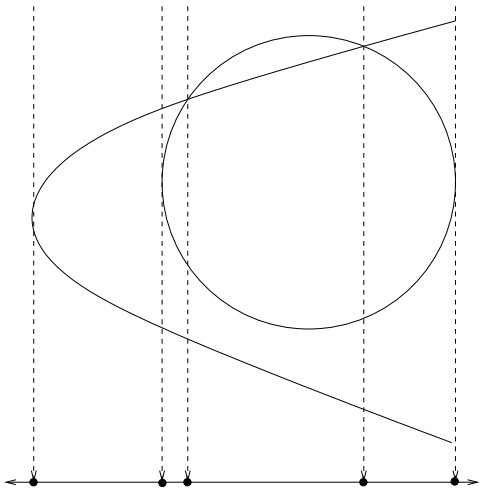

Fig. 10. Two polynomials projected into $\mathbb{R}^1$, preserving the critical points and intersections. After the projection, each interval is lifted into $\mathbb{R}^2$ where it becomes a cylinder of cells. Each new 2-cell is sign-invariant under the polynomials.

completely contained in $\mathcal{C}_{obst}$. Using the connectivity graph of the d-dimensional CAD cells, it is possible to find a collision-free cell path from the cell containing the initial state to the one containing the goal state, if one exists. Schwartz and Sharir then showed how to specify a continuous path from the initial to the goal state that goes from cell to cell in the solution cell path without entering any other cells. The computed path moves from one full-dimensional cell to another, through a connecting cell of one lower dimension; as we discussed above, this is always the case when the free configuration space is taken to be an open set, which is the standard convention. 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 [91] for more details.

B. Algorithm assumptions

To compute smooth vector fields over cylindrical decompositions, our algorithm makes specific input assumptions. First, our algorithm assumes the entire cylindrical algebraic decomposition is specified as input, consisting of all cells (of all levels), together with their corresponding algebraic descriptions. We also assume that the connectivity graph of the decomposition is provided. The CAD algorithm can compute algebraic points for every cell; these will be used as well. Finally, we assume the existence of exact root structure functions for each cell in each level of the decomposition. The root structure function $r : \mathbb{R}^n \times \mathbb{Z}^+$ maps any point $x \in \mathbb{R}^n$ to the set of roots of the polynomials in the $(n+1)$-dimensional lifted cylinder above it (for convenience, consider $\pm\infty$ to be roots). This is a standard part of the CAD algorithm (see [75], [101]), although it is a computationally expensive operation. For some applications, it is possible to improve efficiency by identifying only root *intervals* rather than exact roots, using root separation/gap theorems. See [5], [20].

C. Algorithm Description

We will now present our algorithm for generating smooth feedback plans over CADs. As in Section IV, we will construct smooth feedback over individual cells and then guarantee that smoothness is preserved across the boundaries crossed by the resulting integral curves. We assume that the input to our algorithm is the entire cylindrical algebraic decomposition, consisting of cells of all levels, together with 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 (of any level); 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). As we previously described, 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 open n-dimensional cell C (and its closure $\bar{C}$) and its successor S, both full-dimensional cells of level n. These cells share an $(n-1)$-dimensional face, which we denote F_S.

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. For simplicity, assume that there are no unbounded cells; after describing the algorithm it will become clear that the algorithm works for unbounded cells as well. It is intuitive that each u_i and l_i should correspond to exactly one $(n-1)$-cell (face) in the decomposition, separating C from a neighboring n-cell. However, this is not the case. There may be many faces that correspond to a single bounding polynomial, or none at all. This is illustrated in Figure 11. 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. We use the bar notation to indicate that the logical "face" is the union of a number of actual faces in the decomposition. Note that $\bar{F}_i^+$ corresponds to the zeros of u_i and $\bar{F}_i^-$ corresponds to the zeros of l_i. If the exit face F_S is an upper face corresponding to dimension i, then $F_S \subseteq \bar{F}_i^+$. The two will not generally be equal; F_S may form a hole in the larger face $\bar{F}_i^+$. See Figure 12

We will construct a vector field over $\bar{C}$ in much the same way as we did for convex polytopes. 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 a cell 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

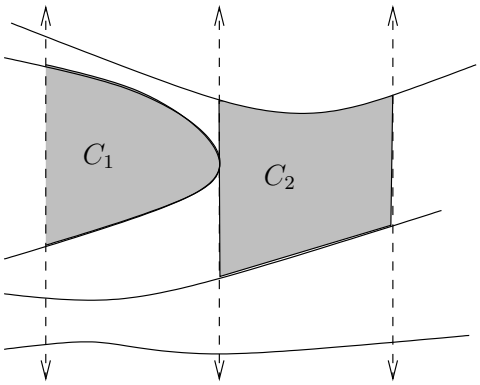

Fig. 11. Adjacent cylinders in a cylindrical algebraic decomposition. Cell C_1 has no adjacent cells corresponding to one of its upper bounding polynomials, and cell C_2 has two adjacent cells corresponding to a lower bounding polynomial (the cells above and below C_1).

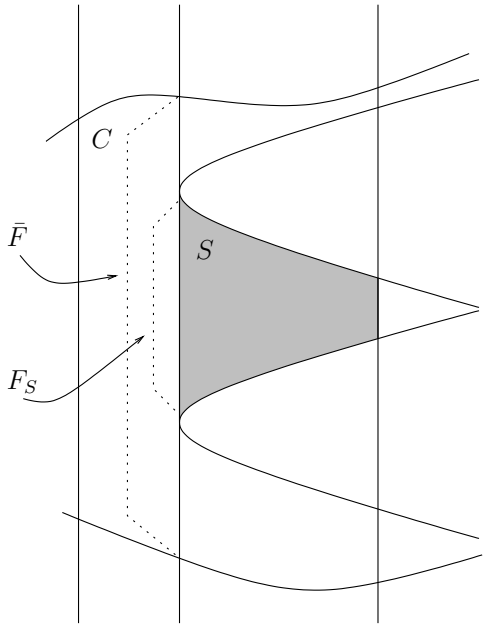

Fig. 12. A cell C, its successor S, and the shared face F_S. The shared face is a hole in the larger face $\bar{F}$ of C. Lifting these cells into higher dimensions could continue to restrict the face they share.

face vector field V_f is defined as

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

This definition ensures that the face vector fields point inward on all faces of C except the exit face, F_S; this implies that no integral curves leave C except by the exit face, as desired.

Recall that through the construction of the CAD, algebraic points have been computed that lie in the interior of each cell; we will use the algebraic points in the cells of level $n-1$, which connect the full-dimensional cells. Denote the algebraic point in F_S as p_a. We now define the *relative height*, which is a diffeomorphism from $\bar{C}$ to the unit cube. For any point $x = (x_1, \ldots, x_n) \in \bar{C}$ and dimension i, let $\mathcal{P}_{i-1}x$ be the projection from the point to its first $i-1$ components, let $r : \mathbb{R}^{i-1} \times \mathbb{Z}^+$ be the root function corresponding to $\mathcal{P}_{i-1}x$, and assume that $r(\mathcal{P}_{i-1}x, l) \leq x_i \leq r(\mathcal{P}_{i-1}x, l+1)$. Define the relative height $h_i : \bar{C} \to [0,1]$ as

$$h_i(x) = \frac{x_i - r(\mathcal{P}_{i-1}x, l)}{r(\mathcal{P}_{i-1}x, l+1) - r(\mathcal{P}_{i-1}x, l)}. \tag{6}$$

This is a smooth mapping, since both the upper and lower bounding polynomials are smooth. Again, even in the case where there are no actual faces corresponding to u_i and l_i, as in Figure 11, the bounding polynomials are well-defined. Also, 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, again as in Figure 11. The lack of smoothness at these points is not problematic because no integral curves of the vector field pass through these points. This is the case because these points are always part of cells that are less than $(n-1)$-dimensional; we have already stated that all integral curves will go from full-dimensional cell to full-dimensional cell through faces of only one dimension less. We can now define the *relative coordinates* of x as $h(x) = (h_1(x), \ldots, h_n(x)) \in [0,1]^n$. We can intuitively define the cell vector field V_c as the vector field which induces a straight line path toward p_a, *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: $V_c(x) := (Jh(x))^{-1}(h(p_a) - h(x))$.

Now, all we need is to define acceptable distance functions to each face. Then, we can blend the component vector fields together as in Section IV, 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. Assume that the exit face $F_S \subseteq \bar{F}_i^{\pm}$; for any face $\bar{F}_j^{\pm}$, $j \neq i$, define the scaled perpendicular distance function $d_\perp$ as follows:

$$d_\perp(x, F) = \begin{cases} \frac{1-h_j(x)}{1-h_j(p_a)} & \text{if } F \subseteq \bar{F}_j^+ \\ \frac{h_j(x)}{h_j(p_a)} & \text{if } F \subseteq \bar{F}_j^- \end{cases} \tag{7}$$

As required, the distance function equals zero on the face itself and is greater than zero elsewhere. Also, at any point that has the same relative height as the algebraic point in a particular dimension, the upper and lower faces will be equidistant. Also, multiple faces that correspond to the same bounding polynomial will have the same distance; this is acceptable, because such faces will have the same face vector field.

In the case of the upper and lower faces in the dimension corresponding to the exit face, a small change must be made. In this case, simply let $d_\perp(x, F) = |h_i(x) - h_i(p)|$. We will also 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 hole 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 a lower level. This means that the "parents" of C and S in the lower level were adjacent i-dimensional cells in the same cylinder, separated by an $(i-1)$-dimensional cell. The parent cells shared a complete face at that level; lifting the

cells into higher dimensions may have restricted the area of the face that they share until they share a face which is a hole in the larger face. An example of this is in Figure 12.

Keeping in mind that each successive lifted dimension adds constraints that may 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 that seems obvious but which is incorrect would be to compute the distance to each of the bounding polynomials of F_S that 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 $x \in \bar{F}_i^{\pm}$. 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 $k \in i+1, \ldots, j-1$; 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 that is uniformly equal to one outside F_S, uniformly zero on some subset $F_S' \subseteq F_S$, and smoothly transitions between the two on $F_S \setminus F_S'$. We need to make several definitions in order to construct this function. Recalling that the cell faces are zeros of polynomials, 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 $u_j^F(z_j^+(x)) = l_j^F(z_j^-(x)) = 0$. For some $\alpha \in (0,1)$ define the *satisfaction function* $w_j : \bar{F}_i^{\pm} \to [0,1]$ as

$$\begin{aligned} w_j(x) = & b\left(\frac{1}{\alpha}\left(\frac{h_j(x) - h_j(p_a)}{h_j(z_j^+(x)) - h_j(p_a)} - (1-\alpha)\right)\right) \\ & + b\left(\frac{1}{\alpha}\left(\frac{h_j(x) - h_j(p_a)}{h_j(z_j^-(x)) - h_j(p_a)} - (1-\alpha)\right)\right). \end{aligned} \tag{8}$$

The satisfaction function w_j considers the bounding polynomials of F_S 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 definition is inductive, as follows:

$$\begin{aligned} \hat{d}_{i+1}(x) &= w_{i+1}(x) \\ &\vdots \\ \hat{d}_j(x) &= \hat{d}_{j-1}(x) + (1 - \hat{d}_{j-1}) w_j(x) \end{aligned} \tag{9}$$

The final distance function, $\hat{d}_n(x)$, is a smooth distance function that can be used to guide the robot through the exit face F_S, which is a "window" in the larger face $\bar{F}_i^{\pm}$. The important results are summarized in the proposition below:

Proposition 3 *The following properties hold:*

1) *For all j such that $i+1 \leq j \leq n$, $\hat{d}_j$ 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 $k \in i+1, \ldots, j-1$. 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 $\hat{F}_i^{\pm} \setminus F_S$. 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 $\hat{F}_i^{\pm}$, 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 in relative coordinates). 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 $\rho(x, \bar{F}_j^{\pm}) \leq \rho(x, \bar{F}_k^{\pm})$, for all k. Second, we say that x lies in the region of influence of F_S if $\rho(x, \bar{F}_i^{\pm}) \leq \rho(x, \bar{F}_k^{\pm})$ 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 $\rho(x, \bar{F}_i^{\pm}) \leq \rho(x, \bar{F}_k^{\pm})$ 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 that 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 Section IV, we use a product of analytic switches. For any face $\bar{F}_j^{\pm}$ with $j \neq i$, use the following:

$$s(p) = 1 - \prod_{\bar{F} \neq \bar{F}_j^{\pm}} \frac{d_\perp(p, \bar{F}) - d_\perp(p, \bar{F}_j^{\pm})}{d_\perp(p, \bar{F})}, \tag{10}$$

in which $\bar{F} \in \mathcal{F}$ are the faces of C. Also, additional product terms need to be added for faces F that share a larger face $\subset \bar{F}_i^{\pm}$ with the exit face F_S. These product terms use $\hat{d}$, rather than $d_\perp$. This function is smooth (except where faces meet), and has the desired property of being identically equal to zero on the face of the cell and one on the boundary of the region of influence. Using the shorthand $b(p) = b(s(p))$, we again define the global vector field V at point p as $V(p) = \mathrm{unit}(b(p)V_f(p) + (1 - b(p))V_c(p))$, in which V_f is the face vector field for the face in whose region of influence p lies, V_c the cell vector field, b the bump function, and unit is the normalization function that forces V to be 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 cell vector field is the vector that points from x to x_g, in relative coordinates. As before, this is defined as $V_c(x) := (Jh(x))^{-1}(h(x_g) - h(x))$. Similarly, the $d_\perp$ function should be modified to consider coordinates relative to the goal point x_g rather than p_a.

D. Formal Analysis

We need to establish that the feedback plan associated with our constructed vector field has all of the required properties; the proofs are similar to those in Section IV.

Theorem 5 *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 (see Section IV). Hence all integral curves of V are smooth. ■

Theorem 6 *The integral curves of V remain in X_{cd}.*

Proof: This property is obvious from the construction of the vector field. In any cell, the face vector field corresponding to the boundary of X_{cd} will be inward pointing, because it 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 the X_{cd} boundary and the integral curves never leave X_{cd}. ■

Theorem 7 *The integral curves of V asymptotically converge to the goal state.*

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 cell vector field is defined as $V_c(x) := (Jh(x))^{-1}(h(p_a) - h(x))$, 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, $h_j(x) \to h_j(p_a), 1 \le j \le n$. 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 must 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 cell vector field and face vector fields all cause the relative coordinates to converge to those of the goal state. 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 any neighborhood of the goal state in finite time, we have the global result that all integral curves of V asymptotically converge to the goal state. ■

VI. Conclusion and Future Work

We have presented algorithms that construct smooth feedback plans on two different types of cell decompositions of any finite dimension. A smooth feedback plan defines a feedback law, which smoothly stabilizes the system to a selected goal state from anywhere in the state space, while avoiding obstacles. To accomplish this, we construct a vector field on the state space that is smooth except for a set of measure zero and which has integral curves that are smooth and converge to the goal. Feedback (i.e., closed loop control) is important because it provides a measure of robustness to uncertainty in sensing and control. Using feedback, inputs are computed as (typically simple) functions of the current state, and it is therefore possible to compute them at a very high rate, further improving performance. Smooth feedback also contributes to practical robustness because the control changes smoothly in response to small perturbations in the state. This implies that when using smooth feedback, a practical system is likely to perform better when errors occur. Although smooth controls can still have arbitrarily high derivatives (any continuous curve has a smooth approximation that can be arbitrarily close), they tend to avoid rapid input changes whenever possible, also leading to more reliable operations.

We first described how to construct smooth feedback plans for a point robot moving in a state space with a piecewise linear boundary. We did this by first computing a decomposition

of the state space into convex polytopes, and then constructing local feedback laws over individual cells and smoothly transitioning between them. We also described how to construct smooth feedback plans over the cells of a cylindrical algebraic decomposition. The important work of Schwartz and Sharir demonstrated how to solve the general piano movers' problem using the Collins decomposition, a type of cylindrical algebraic decomposition [91]. Our work extends this, showing that not only can these decompositions solve single motion planning queries, but can also be used to compute a global smooth feedback plan. Moreover, this feedback can be computed essentially for free, given the complexity of the CAD cells. Hence, our work proves the existence of smooth feedback plans for a very general class of motion planning problems. To our knowledge, this is the most general result to date.

Although this is an interesting theoretical result, it has limited practical utility because the number of cells in a Collins decomposition is doubly exponential in the dimension of the space. Since our algorithm is applicable to any type of cylindrical algebraic decomposition, our method may be practical for problems that use CADs but do not require the full complexity of Collins decompositions. For example, our approach applies to the specific decompositions produced for a ladder robot [3] and a robot translating and rotating in the plane [2].

In conclusion, we have presented algorithms that efficiently solve the feedback motion planning problem, once a cell decomposition has been computed. This result builds on existing complete path planning methods and obtains a smooth vector field for virtually no additional cost. However, two shortcomings of the method remain as topics of future research. First, there is no explicit way to induce a preference toward trajectories that have lower curvature. We simply provide smoothness, which in the worst case could be arbitrarily close to a non-smooth point. Second, our methods produce trajectories that are sensitive to the particular cell decomposition, rather than focusing on the quality of the overall trajectories. Computing vector fields that optimize quality measures while maintaining smoothness would be ideal; however, this seems intractable given our present understanding of the state of the art.

ACKNOWLEDGMENTS

This work was funded in part by an NSF Graduate Research Fellowship and NSF Award 0535007. The authors appreciate the helpful insights and suggestions offered by the reviewers, which greatly improved this paper.

REFERENCES

[1] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[2] 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 and Automation*, pages 1656–1660, 1988.

[3] J. Bañon. Implementation and extension of the ladder algorithm. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1548–1553, 1990.

[4] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research*, 10(6):628–649, December 1991.

[5] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry*. Springer-Verlag, Berlin, 2003.

[6] C. Belta, L. C. G. J. M. Habets, and V. Kumar. Control of multi-affine systems on rectangles with applications to hybrid biomolecular networks. In *Proceedings IEEE Conference on Decision and Control*, pages 534–539, 2002.

[7] C. Belta, V. Isler, and G. J. Pappas. Discrete abstractions for robot motion planning and control in polygonal environments. *IEEE Transactions on Robotics*, 21(5):864–874, October 2005.

[8] A. Bemporad and M. Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35:407–427, 1999.

[9] D. Bertsekas. *Dynamic Programming and Optimal Control: Volume I*. Athena Scientific, Belmont, MA, 2000.

[10] J. E. Bobrow, S. Dubowsky, and J. S. Gibson. Time-optimal control of robotic manipulators along specified paths. *International Journal of Robotics Research*, 4(3):3–17, 1985.

[11] R. Bohlin. *Robot Path Planning*. PhD thesis, Chalmers University, Gothenburg, Sweden, 2002.

[12] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, 1989.

[13] J. Borenstein and Y. Koren. The vector field histogram – fast obstacle avoidance for mobile robots. *IEEE Transactions on Robotics and Automation*, 7(3):278–288, 1991.

[14] M. Branicky. Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):475–482, 1998.

[15] M. Branicky. A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1):31–45, January 1998.

[16] M. S. Branicky, V. S. Borkar, and S. K. Mitter. A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1):31–45, 1998.

[17] O. Brock and O. Khatib. High-speed navigation using the global dynamic window approach. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 341–346, 1999.

[18] W. L. Brogan. *Modern Control Theory*. Prentice Hall, Upper Saddle River, NJ, third edition, 1991.

[19] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *International Journal of Robotics Research*, 18(6):534–555, 1999.

[20] J. F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1988.

[21] I. Cervantes, R. Kelly, J. Alvarez-Ramirez, and J. Moreno. A robust velocity field control. *IEEE Transactions on Control Systems Technology*, 10(6):888–894, November 2002.

[22] B. Chazelle. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry*, 6(5):485–524, 1991.

[23] C.-T. Chen. *Linear System Theory and Design*. Oxford University Press, New York, third edition, 1999.

[24] H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, Cambridge, MA, 2005.

[25] P. Choudhury and K. Lynch. Trajectory planning for kinematically controllable underactuated mechanical systems. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, pages 559–576, 2002.

[26] G. E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In *Proceedings Second GI Conference on Automata Theory and Formal Languages*, pages 134–183, Berlin, 1975. Springer-Verlag.

[27] D. C. Conner, H. Choset, and A. A. Rizzi. Provably correct navigation and control for non-holonomic convex-bodied systems operating in cluttered environments. In *Proceedings Robotics: Science and Systems*, pages 57–64, 2006.

[28] 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*, pages 3546–3551, 2003.

[29] C. Connolly and R. Grupen. The application of harmonic potential functions to robotics. *Journal of Robotic Systems*, 10(7):931–946, 1993.

[30] C. Connolly, K. Souccar, and R. Grupen. A Hamiltonian framework for kinodynamic planning. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2746–2751, 1995.

[31] C. I. Connolly. Applications of harmonic functions to robotics. In *IEEE Symposium on Intelligent Control*, pages 498–502, 1992.

[32] C. I. Connolly, J. B. Burns, and R. Weiss. Path planning using Laplace's equation. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2102–2106, May 1990.

[33] W. E. Dixon, T. Galluzo, G. Hu, and C. Crane. Adaptive velocity field control of a wheeled mobile robot. In *IEEE Fifth International Workshop on Robot Motion and Control*, pages 145–150, 2005.

[34] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas. Hybrid controllers for path planning: A temporal logic approach. In *Proceedings IEEE Conference on Decision and Control, and the European Control Conference*, pages 4885–4890, 2005.

[35] P. Ferbach. A method of progressive constraints for nonholonomic motion planning. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2949–2955, 1996.

[36] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4:23–33, 1997.

[37] J. E. Goodman and J. O'Rourke, Eds. *Handbook of Discrete and Computational Geometry*. Chapman and Hall/CRC Press, New York, second edition, 2004.

[38] L. C. G. J. M. Habets, P. J. Collins, and J. H. van Schuppen. Reachability and control synthesis for piecewise-affine hybrid systems on simplices. *IEEE Transactions on Automatic Control*, 51(6):938–948, 2006.

[39] L. C. G. J. M. Habets and J. H. van Schuppen. Control of piecewise-linear hybrid systems on simplices and rectangles. In M. D. Di Benedetto and A. Sangiovanni-Vincentelli, editors, *Hybrid Systems: Computation and Control*, pages 261–274. Springer-Verlag, Berlin, 2001.

[40] L. C. G. J. M. Habets and J. H. van Schuppen. A control problem for affine dynamical systems on a full-dimensional polytope. *Automatica*, 40:21–35, 2004.

[41] L. C. G. J. M. Habets and J. H. van Schuppen. Control to facet problems for affine systems on simplices and polytopes – with applications to control of hybrid systems. In *Proceedings IEEE Conference on Decision and Control, and the European Control Conference*, pages 4175–4180, 2005.

[42] D. Halperin. Arrangements. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 529–562. Chapman and Hall/CRC Press, New York, second edition, 2004.

[43] T. Hebert. *Navigation of an Autonomous Vehicle Using a Combined Electrostatic Potential Field/Fuzzy Inference Approach*. PhD thesis, Univ. Southwestern Louisiana, 1998.

[44] S. Hertel and K. Mehlhorn. Fast triangulation of simple polygons. In *Proceedings of the International FCT-Conference on Fundamentals of Computation Theory*, pages 207–218, 1983.

[45] D. Hsu, R. Kindel, J-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, 21(3):233–255, 2002.

[46] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *International Journal Computational Geometry and Applications*, 4:495–512, 1999.

[47] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[48] H. K. Khalil. *Nonlinear Systems*. Macmillan, New York, 2002.

[49] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[50] D. G. Kirkpatrick. Optimal search in planar subdivisions. *SIAM Journal of Computing*, 12:28–35, 1983.

[51] M. Kloetzer and C. Belta. Reachability analysis of multi-affine systems. In J. Hespanha and A. Tiwari, editors, *Hybrid Systems: Computation and Control*, pages 348–362. Springer-Verlag, Berlin, 2006.

[52] N. Y. Ko and R. G. Simmons. The lane-curvature method for local obstacle avoidance. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1615–1621, 1998.

[53] B. H. Krogh. A generalized potential field approach to obstacle avoidance control. In *Proceedings of SME Conference on Robotics Research*, pages 14–19, August 1984.

[54] A. Ladd and L. E. Kavraki. Fast tree-based exploration of state space for robots with dynamics. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, pages 297–312, Zeist, The Netherlands, 2004.

[55] F. Lamiraux and J.-P. Laumond. Flatness and small-time controllability of multibody mobile robots: Application to motion planning. *IEEE Transactions on Automatic Control*, 45(10):1878–1881, 2000.

[56] J.-C. Latombe. *Robot Motion Planning*. Kluwer, Boston, MA, 1991.

[57] J.-P. Laumond, S. Sekhavat, and F. Lamiraux. Guidelines in nonholonomic motion planning for mobile robots. In J.-P. Laumond, editor, *Robot Motion Planning and Control*, pages 1–53. Springer-Verlag, Berlin, 1998.

[58] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006.

[59] S. M. LaValle and P. Konkimalla. Algorithms for computing numerical optimal feedback motion strategies. *International Journal of Robotics Research*, 20(9):729–752, September 2001.

[60] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.

[61] S. M. LaValle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In B. R. Donald, K. M. Lynch, and D. Rus, editors, *Algorithmic and Computational Robotics: New Directions*, pages 293–308. A K Peters, Wellesley, MA, 2001.

[62] P. Y. Li and R. Horowitz. Passive velocity field control of mechanical manipulators. *IEEE Transactions on Robotics and Automation*, 15(4):751–763, August 1999.

[63] P. Y. Li and R. Horowitz. Passive velocity field control (PVFC): Part I–Geometry and robustness. *IEEE Transactions on Automatic Control*, 46(9):1346–1359, September 2001.

[64] P. Y. Li and R. Horowitz. Passive velocity field control (PVFC): Part II–Application to contour following. *IEEE Transactions on Automatic Control*, 46(9):1360–1371, September 2001.

[65] D. Liberzon. *Switching in Systems and Control*. Birkhäuser, Boston, MA, 2003.

[66] 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*, pages 2406–2411, 2006.

[67] S. R. Lindemann and S. M. LaValle. Smoothly blending vector fields for global robot navigation. In *Proceedings IEEE Conference Decision & Control*, pages 3353–3559, 2005.

[68] S. R. Lindemann and S. M. LaValle. Computing smooth feedback plans over cylindrical algebraic decompositions. In *Proceedings Robotics: Science and Systems*, 2006.

[69] S. R. Lindemann and S. M. LaValle. A multiresolution approach for motion planning under differential constraints. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 139–144, 2006.

[70] S. R. Lindemann and S. M. LaValle. Smooth feedback for car-like vehicles in polygonal environments. In *IEEE International Conference on Robotics & Automation*, pages 3104–3109, 2007.

[71] S. Loizou and K. Kyriakopoulos. A feedback-based multiagent navigation framework. *International Journal of Systems Science*, 37(6):377–384, 2006.

[72] S. G. Loizou, D. V. Dimarogonas, and K. J. Kyriakopoulos. Decentralized feedback stabilization of multiple nonholonomic agents. In *Proceedings IEEE International Conference on Robotics and Automation*, 2004.

[73] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic synthesis of fine-motion strategies for robots. *International Journal of Robotics Research*, 3(1):3–24, 1984.

[74] S. A. Masoud and A. A. Masoud. Constrained motion control using vector potential fields. *IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans*, 30(3):251–272, May 2000.

[75] B. Mishra. *Algorithmic Algebra*. Springer-Verlag, New York, 1993.

[76] B. Mishra. Computational real algebraic geometry. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 537–556. CRC Press, New York, 1997.

[77] I. M. Mitchell and S. Sastry. Continuous path planning with multiple constraints. In *Proceedings IEEE Conference on Decision and Control*, pages 5502–5507, 2003.

[78] J. Moreno and R. Kelly. Hierarchical velocity field control for robot manipulators. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 4374–4379, 2003.

[79] R. M. Murray and S. Sastry. Nonholonomic motion planning: Steering using sinusoids. *IEEE Transactions on Automatic Control*, 38(5):700–716, 1993.

[80] A. Narkhede and D. Manocha. Fast polygon triangulation based on seidel's algorithm. In A. Paeth, editor, *Graphics Gems V*, pages 394–397. Academic, New York, 1995.

[81] C. O'Dunlaing, M. Sharir, and C. K. Yap. Retraction: A new approach to motion planning. In J .T .Schwartz, M. Sharir, and J. Hopcroft, editors, *Planning, Geometry, and Complexity of Robot Motion*, pages 193–213. Ablex, Norwood, NJ, 1987.

[82] P. Ogren and N. E. Leonard. A convergent dynamic window approach to obstacle avoidance. *IEEE Transactions on Robotics*, 21(2):188–195, 2005.

[83] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proceedings IEEE Symposium on Foundations of Computer Science*, pages 421–427, 1979.

[84] A. Richards, J. How, T. Schouwenaars, and E. Feron. Spacecraft trajectory planning with avoidance constraints using mixed-integer linear programming. *AIAA Journal on Guidance, Control and Dynamics*, 25(4):755–764, 2002.

[85] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential fields. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, October 1992.

[86] A. A. Rizzi. Hybrid control as a method for robot motion programming. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 832–837, 1998.

[87] B. Roszak and M. E. Broucke. Necessary and sufficient conditions for reachability on a simplex. In *Proceedings IEEE Conference on Decision and Control, and the European Control Conference*, pages 4706–4711, 2005.

[88] P. F. Rowat. *Representing the Spatial Experience and Solving Spatial Problems in a Simulated Robot Environment*. PhD thesis, University of British Columbia, 1979.

[89] S. Sastry. *Nonlinear Systems: Analysis, Stability, and Control*. Springer-Verlag, Berlin, 1999.

[90] T. Schouwenaars, J. P. How, and E. Feron. Receding horizon path planning with implicit safety guarantees. In *Proceedings IEEE American Control Conference*, pages 5576–5581, 2004.

[91] J. T. Schwartz and M. Sharir. On the piano movers' problem: II. General techniques for computing topological properties of algebraic manifolds. *Advances in Applied Mathematics*, 12:298–351, 1983.

[92] R. Seidel. A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons. *Computational Geometry: Theory and Applications*, 1(1):51–64, 1991.

[93] 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*, 17:840–857, 1998.

[94] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, 93(4):1591–1595, 1996.

[95] J. A. Sethian and A. Vladimirsky. Ordered upwind methods for static Hamilton-Jacobi equations: Theory and algorithms. *SIAM Journal on Numerical Analysis*, 41(1):325–363, 2003.

[96] J. R. Shewchuk. Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator. In Ming C. Lin and Dinesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148, pages 203–222. Springer-Verlag, Berlin, 1996.

[97] K. G. Shin and N. D. McKay. Minimum-time control of robot manipulators with geometric path constraints. *IEEE Transactions on Automatic Control*, 30(6):531–541, 1985.

[98] R. Simmons. The curvature-velocity method for local obstacle avoidance. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 3375–3382, 1996.

[99] C. Stachniss and W. Burgard. An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 508–513, 2002.

[100] M. R. Stan, W. P. Burleson, C. I. Connolly, and R. A. Grupen. Analog VLSI for robot path planning. *Journal of VLSI Signal Processing*, 8(1):61–73, 1994.

[101] A. Strzebonski. Cylindrical algebraic decomposition using validated numerics. *Journal of Symbolic Computation*, 41:1021–1038, 2006.

[102] H. Tanner and A. Kumar. Towards decentralization of multi-robot navigation functions. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 4143–4148, 2005.

[103] H. G. Tanner, S. G. Loizou, and K. J. Kyriakopoulos. Nonholonomic navigation and control of cooperating mobile manipulators. *IEEE Transactions on Robotics and Automation*, 19(1):53–64, February 2003.

[104] L. Tarassenko and A. Blake. Analogue computation of collision-free paths. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 540–545, 1991.

[105] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, September 1995.

[106] I. Ulrich and J. Borenstein. VFH*: Local obstacle avoidance with look-ahead verification. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2505–2511, 1991.

[107] I. Ulrich and J. Borenstein. VFH+: Reliable obstacle avoidance for fast mobile robots. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 1572–1577, 1998.

[108] K. P. Valavanis, T. Hebert, R. Kolluru, and N. Tsourveloudis. Mobile robot navigation in 2-D dynamic environments using an electrostatic potential field. *IEEE Transactions on Systems, Man, and Cybernetics–Part A: Systems and Humans*, 30(2):187–196, 2000.

[109] A. van der Schaft and H. Schumacher. *An Introduction to Hybrid Dynamical Systems*. Springer, London, 2000.

[110] R. Volpe and P. Khosla. Manipulator control with superquadric artificial potential functions: Theory and experiments. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(6):1423–1436, 1990.

[111] Y. Wang and G. S. Chirikjian. A new potential field method for robot path planning. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 977–982, 2000.

[112] S. Waydo and R. M Murray. Vehicle motion planning using stream functions. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2484–2491, 2003.

[113] M. Yamakita, K. Suzuki, X.-Z. Zheng, M. Katayama, and K. Ito. An extension of passive velocity field control to cooperative multiple manipulator systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 11–16, 1997.

[114] M. Yamakita, T. Yazawa, X.-Z. Zheng, and K. Ito. An application of passive velocity field control to cooperative multiple 3-wheeled mobile robots. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 368–373, 1998.

[115] L. Yang and S. M. LaValle. The sampling-based neighborhood graph: A framework for planning and executing feedback motion strategies. *IEEE Transactions on Robotics and Automation*, 20(3):419–432, 2004.

[116] Y. Yang and O. Brock. Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation. In *Proceedings Robotics: Science and Systems*, pages 281–288, 2006.

[117] Y. Zhang. *Sensor-Based Potential Field Method for Robot Motion Planning*. PhD thesis, Univ. Southwestern Louisiana, 1995.