© 2008 Stephen R. Lindemann

SMOOTH FEEDBACK PLANNING

BY

STEPHEN R. LINDEMANN

B.A., Covenant College, 2000
B.S., University of Kentucky, 2001
M.S., University of Illinois at Urbana-Champaign, 2005

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2008

Urbana, Illinois

Doctoral Committee:

Professor Mark W. Spong, Chair
Associate Professor Steven M. LaValle, Director of Research
Professor Sean P. Meyn
Professor Seth A. Hutchinson
Assistant Professor Dusan M. Stipanovic
Associate Professor Magnus B. Egerstedt, Georgia Institute of Technology

ABSTRACT

The contribution of this dissertation is the presentation of a new algorithm for the construction of feedback controllers with global convergence, safety, and smoothness guarantees. This algorithm integrates motion planning's emphasis on collision avoidance and algorithmic completeness with control theory's insistence on the use of feedback to achieve robust, efficient, real time control. This combination of features renders this algorithm uniquely applicable to real world robot navigation problems.

Historically, the motion planning problem is to compute a continuous, collision-free path between given initial and goal configurations, or to return that no such path exists. These algorithms typically assume both perfect sensing and perfect control; i.e., the environment and the robot's configuration are perfectly known at all times, and the robot's motion is perfectly predictable. From a practical point of view, however, neither of these assumptions is actually valid. Even if a collision-free path can be computed, the robot cannot follow it exactly; once it has deviated from the precomputed trajectory, what should the robot do? Questions like these are a motivation for feedback control, in which a vector field is computed, rather than a single path. Since the vector field is defined everywhere, the robot is always able to follow it. This implies that a navigation system based on feedback is more robust to errors in sensing and control than one using an open loop trajectory, even though both approaches still assume perfect sensing and control from a theoretical point of view. Traditionally, feedback control has dealt exclusively

with environments free from obstacles. A number of attempts have been made to compute feedback control laws for systems in the presence of obstacles, but they typically suffer from either lack of completeness guarantees or limited ability to be applied in practice. In contrast, the family of algorithms described in this dissertation provide strong completeness guarantees and can be applied easily to a variety of frequently encountered robot navigation problems.

Algorithms, theoretical analysis, and practical examples (as applicable) are presented for three main problems, and several variations thereon. First, the problem of a fully actuated point robot moving in a piecewise linear environment of arbitary dimension is addressed. Second, fully actuated robots with bodies are considered—both practical problems like planar disc or polygonal robots, and the fully general case of a semi-algebraic robot moving amidst semi-algebraic obstacles (the generalized piano mover's problem). Third, feedback plans for point robots with simple nonholonomic constraints, such as unicycles and car-like robots, are described. Each of these problems is addressed using variations on a unified approach. First, the environment is decomposed into simple (often convex) cells; second, local vector fields are created and combined together to create a solution to the global navigation problem. In addition to standard navigation problems, approaches to trajectory tracking and multiple robot coordination are described to illustrate the flexibility of the approach.

To Amanda

ACKNOWLEDGMENTS

This work would not have been possible without encouragement and support of countless family, friends, and colleagues. I can by no means mention them all here; I only hope to give recognition to a few individuals who have deeply impacted me during my studies at the University of Illinois.

I would like to thank my adviser, Steve LaValle, for his mentorship throughout my graduate studies. We met during my first several weeks at the University of Illinois, and I count myself extremely lucky to have been able to work with him for the past six years. His dedication to "truth and scholarship" and persistent inquisitiveness have all shaped my intellectual growth and perspective.

I am as impressed and inspired by the quality of the faculty at UIUC as I was during my first semester there. I have had many gifted instructors, all of whom deserve my thanks. Many classes were difficult yet immensely rewarding, such as control theory with Tamer Basar and differentiable manifolds with Rob Ghrist. I also owe a great debt to all members of my Ph.D. committee: Sean Meyn, Mark Spong, Seth Hutchinson, Dusan Stipanovic, and Magnus Egerstedt (Georgia Tech). Thank you for granting me your time and advice during my work on this dissertation.

I am grateful for the friendship and intellectual stimulation of my fellow students in Steve LaValle's group: Peng Cheng, Jason O'Kane, Shai Sachs, Benjamin Tovar, Anna Yershova, Hamid Chitsaz, George Zaimes, and Jingjin Yu. I would especially like to thank Jason O'Kane, with whom I shared an office for several

years. Having a foosball table in your office is one of the joys of being a graduate student.

In addition to my many professional colleagues, my family's support has been invaluable in giving me the strength and determination to complete this dissertation. I would like to thank my wife, Amanda, for her patience as so much of my time and energy has been absorbed by this project. I will always remember our years as students in Champaign with great fondness.

I owe a tremendous debt to my parents, whose love and training still guide me. Thank you for your many sacrifices on my behalf. I also would like to thank my brother, David, for being such a good friend over the years. I greatly appreciate your competitiveness, acute intellect, and (even!) your sense of humor.

I have benefitted from a great many friendships over the years. I would especially like to thank Josh Coe and Nelson Nguyen for many exhausting racquetball matches, Mike Shea and Jeremy Hatcher for some brilliant darts games, and the Johnsons for their friendship and hospitality every Sunday evening. I am deeply grateful to you all.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Robots in the Real World

By many accounts, robotics could be the next big thing. Millions of Roomba vacuuming robots have been sold to date [1]; robots are being deployed together with soldiers for a variety of applications [2, 3]; they pour drinks [4]; they take care of the elderly [5]. Microsoft founder Bill Gates is so convinced that robotics will become important that he wanted Microsoft to participate [6]; the company recently released its Robotics Studio, a platform (freely available for noncommercial use) designed to make it easy to design robots using common sensors and components, write algorithms to control them, and simulate them in realistic virtual environments [7]. DARPA has sought to stimulate research in autonomous vehicles and driving through a series of "Grand Challenge" competitions. The challenge in the first two competitions was to navigate through a 142-mile desert course in less than 10 hours; no one was successful in the first year, but three teams completed the course in the second year. The third competition, the DARPA Urban Challenge, focused on safe driving in urban environments; several competitors successfully completed the challenge.

Many fields have claimed that significant breakthroughs were just around the corner, and that society would be revolutionized by the innovations that were ready to take place. While these changes are sometimes realized (the Internet, for example), it is far more common that the promised breakthroughs never materi-

alize. What will it take for the promise and potential of robotics to be realized, rather than remaining perpetually just out of reach? Building robots has never been easier, computing power is astonishingly cheap, battery technology continues to improve (due in part to the push for energy efficient vehicles), and sensors are cheaper and more reliable than ever. What, then, remains?

In nearly all of the areas where robotics is poised to have an impact, robots will be required to operate outside of the highly structured settings in which they have historically been successful. They must navigate in large environments with many different kinds of obstacles; they must interact safely with other autonomous, unpredictable agents who are also moving in that environment, such as people (not to mention their pets). Robust, efficient algorithms—for robot control, for navigation, for dealing with uncertainties in the environment and in sensing and control—are the key ingredients that will enable robots to successfully solve tasks in these environments.

This dissertation presents a set of algorithms targeted at the problem of robust navigation in obstacle-cluttered environments. More precisely, these algorithms compute feedback controllers that guarantee obstacle avoidance and global convergence to a goal state. They bring together elements of classical feedback control and algorithmic motion planning, both of which are extremely important for robot navigation. Previous attempts to integrate motion planning and control frequently are based on potential functions; in contrast, the algorithms presented in this dissertation rely on constructing feedback through careful combinations of local vector fields. In practice, it is much easier to define these vector fields than to construct potential fields offering the same safety and convergence guarantees. In addition to solving problems of theretical interest, these algorithms are also useful for a variety of practical problems.

The remainder of this introduction will be devoted to outlining basic concepts in feedback control and motion planning and showing how they are both essential to building robot systems that can solve challenging problems in the real world. The formulation of the general problem of smooth feedback planning will be presented. Finally, the results presented in subsequent chapters will be described.

1.2 Feedback and Motion Planning

In this section, a very high level overview of algorithmic motion planning and feedback control is presented, along with their importance for robust robot navigation. Historical ideas and approaches that have had an impact on the research presented here are emphasized. First, a formal characterization of the motion planning problem and overview the development of the field is presented. Second, basic concepts in control theory, its relation to algorithmic motion planning, and the importance of feedback control for practical robotics problems are described.

1.2.1 Algorithmic motion planning

The motion planning problem is to compute a collision-free trajectory between two points for a robot moving in the world. Before formalizing this description, we need to define a number of important items. First, let the *world* be $\mathcal{W} = \mathbb{R}^2$ or $\mathcal{W} = \mathbb{R}^3$. The world contains an *obstacle region*, which is a closed semi-algebraic set $\mathcal{O} \subset \mathcal{W}$. Let the *robot* likewise be a closed semi-algebraic set $\mathcal{A} \subset \mathcal{W}$.[1] All the ideas in this section can be extended to the case where the robot consists of a chain of bodies; however, we will discuss only the single rigid body case.

The configuration space $\mathcal{C}$ of the robot is the manifold corresponding to the space of transformations of the robot. The number of "degrees of freedom" of

[1]The formulation in this section largely follows [8], which contains much more detail.

the robot is then equal to the dimension of the configuration space. For example, a robot that can only translate in the plane ($\mathcal{W} = \mathbb{R}^2$) has only two degrees of freedom, so $\mathcal{C} = \mathbb{R}^2$. A robot translating and rotating in three-dimensional space has six degrees of freedom, three for translation and three for rotation. In this case, $\mathcal{C} = \mathbb{R}^3 \times \mathrm{SO}(3) = \mathrm{SE}(3)$.

The obstacles in the world induce an *obstacle region* in the configuration space, $\mathcal{C}_{obst} \subseteq \mathcal{C}$. For some configuration $q \in \mathcal{C}$, let T_q be the transformation corresponding to q. Let

$$\mathcal{A}(q) = \bigcup_{a \in \mathcal{A}} T_q(a) \tag{1.1}$$

be the robot body transformed by T_q. The the obstacle region is defined as

$$\mathcal{C}_{obst} = \{q \in \mathcal{C} | \mathcal{A}(q) \cap \mathcal{O} \neq \emptyset\}. \tag{1.2}$$

The *free configuration space* $\mathcal{C}_{free}$ is then defined as $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obst}$. Note that since the robot and obstacles are closed sets, $\mathcal{C}_{obst}$ is also closed. Therefore, $\mathcal{C}_{free}$ is an open set.

We can now give a precise definition of the motion planning problem. When the robot's motion has no differential constraints, this is referred to as the *piano mover's problem*.

Definition 1.1 (The Piano Mover's Problem)

Given:

1. *A world $\mathcal{W}$, obstacles $\mathcal{O}$, and robot $\mathcal{A}$;*
2. *A corresponding configuration space $\mathcal{C}$, partitioned into $\mathcal{C}_{free}$ and $\mathcal{C}_{obst}$;*
3. *An initial state $q_i \in \mathcal{C}_{free}$ and a goal state $q_g \in \mathcal{C}_{free}$.*

An algorithm solves the piano mover's problem if for all choices of the above, it either returns a continuous path $\tau : [0,1] \to \mathcal{C}_{free}$ such that $\tau(0) = q_i$, $\tau(1) = q_g$, or correctly reports that no such path exists, in finite time.

If an algorithm solves the piano mover's problem as given above it is said to be *complete*. Two weaker forms of solutions exist: *resolution completeness* and *probabilistic completeness*. An algorithm is resolution complete if it is guaranteed to find a solution in finite time, if one exists. An algorithm is probabilistically complete if the probability of finding a solution if one exists goes to one as time goes to infinity, if a solution exists. If there is no solution, resolution complete and probabilistically complete algorithms are not required to report this in finite time.

In the problem above, the robot is said to be *free-flying* because there are no constraints on its motion. In reality, however, there are almost always motion constraints, which can be expressed as a set of differential equations. As a result, the configuration space is no longer adequate to fully describe the system; the *state space* must be considered. Intuitively, the state space X is the space of minimal dimension such that all differential equations describing the dynamics of the system are first-order (i.e., they depend on the state x and derivative $\dot{x}$, but not $\ddot{x}$). Under very general conditions, the constraints can be written as a *state transition equation* $\dot{x} = f(x, u)$, in which u is the control input. Existence and uniqueness of solutions to differential equations guarantees that given an initial state x_0 and control $u_{[0,t)}$, the state at time t, $x(t)$, exists and is unique.

In addition to equality constraints, there may also be inequality constraints involving state space variables which do not arise from obstacles in the world. If there are k such constraints, they can be written as $g_i(x) < 0$ for $1 \leq i \leq k$. An example of such a constraint is a maximum velocity constraint. These constraints, together with the world obstacle constraints, lead to a definition of X_{obst}. Let

$\kappa : X \to \mathcal{C}$ be the function giving the configuration q corresponding a particular state x. The obstacle region is then defined as

$$X_{obst} = \{x \in X | \kappa(x) \in \mathcal{C}_{free},\, g_i(x) \leq 0,\, \forall i \in 1, \ldots, k\}. \tag{1.3}$$

As expected, the free state space is then $X_{free} = X \setminus X_{obst}$. We can now formulate the problem of motion planning under differential constraints (PDC):

Definition 1.2 (Motion Planning Under Differential Constraints) *Given:*

1. *A world $\mathcal{W}$, obstacles $\mathcal{O}$, robot $\mathcal{A}$, and configuration space $\mathcal{C}$;*
2. *A smooth manifold X, the state space;*
3. *An input space U;*
4. *A state transition equation: $\dot{x} = f(x, u),\ \forall x \in X$;*
5. *An initial state $x_i \in X_{free}$ and a goal set $x_g \subset X_{free}$.*

An algorithm solves the problem of motion planning under differential constraints if for all choices of the above, it either returns an input trajectory $\tilde{u} : [0, t_f) \to U$ for some $t_f \in [0, \infty)$, such that $x(s) = x_i + \int_0^s f(x, \tilde{u}(t))dt \in X_{free}\ \forall s \in (0, t_f)$, and $x(t_f) \in x_g$, or correctly reports that no such path exists, in finite time.

Complete, resolution complete, and probabilistically complete algorithms are as defined before.

In PDC, the problem is no longer as simple as computing a collision-free path through the configuration space (as though that could be called simple!). The reason for this is that in general, it is difficult—sometimes even impossible—to transform an arbitrary path in the configuration space to a corresponding one that

satisfies the state transition equation in the state space, although this is possible for certain special classes of systems. Therefore, the PDC problem requires that complete algorithms compute *input* trajectories (or *controls*) that induce state trajectories that take the robot from the given initial state to the goal set, while statisfying the constraints (obstacle constraints and motion constraints).

Secondly, in PDC the solution (state) trajectory is required to terminate in a goal set, rather than a goal state. If the robot must go from a single initial point to a single goal point, the problem is a classic two-point boundary problem, which is extremely difficult to solve, and even eliminates the ability to use some types of algorithms, such as resolution complete algorithms. The relaxation that the goal is allowed to be an open set permits the use of efficient search algorithms.

In this work, two simple PDC problems are considered. We will present smooth feedback planning algorithms for unicycles and car-like robots, which are both types of first-order nonholonomic systems. The are first order, meaning that the velocity of the robots can be directly controlled, as opposed to indirectly through acceleration inputs. They are also nonholonomic; the differential constraints on their motion do not permit them to locally move in any direction. These systems will be described in more detail in Chapter 3.

1.2.2 Approaches to motion planning

Motion planning as an area of active research can be traced to the introduction of the configuration space idea [9, 10], as described above. Reif originally proved the PSPACE hardness of motion planning [11]. Prior to the introduction of the configuration space approach, motion planning considered only the workspace, using approaches such as swept volumes to determine if a path was feasible. With configuration spaces, the problem might seem considerably simplified, since the

robot is reduced to a point in the configuration space. While the configuration space is the most natural way to approach the robot path planning problem, it has two important drawbacks. First, the dimension of the space increases; instead of dealing with objects in only two or three dimensions, planning in the configuration space involves computing a continuous, collision-free trajectory in a high-dimensional space. Second, the obstacle region can be very complicated and difficult to explicitly compute. This drawback in particular proved to be a significant challenge in the early years of motion planning research.

Most motion planning research in the 1980s involved computing an explicit representation of $\mathcal{C}_{obst}$ and $\mathcal{C}_{free}$. The first general, complete motion planning algorithm for a free-flying robot in three dimensions was given by Schwartz and Sharir [12]. Their algorithm was based on the Collins decomposition [13], a type of cylindrical algebriac decomposition; the running time was doubly exponential in the dimension of the configuration space. Canny's roadmap algorithm improved the time bound to "only" singly exponential [14]. Both of these techniques rely on algorithms in real algebraic geometry, which are very difficult to implement in practice [15]. Other algorithms which relied, either completely or in part, on computing a representation of $\mathcal{C}_{obst}$ include the planner by Brooks and Lozano-Pérez for a polygon rotating and translating in the plane [16], work by Donald for planning for a 3D rigid body [17, 18], and a planner by Lozano-Pérez for manipulator arms [19]. References to many combinatorial planners and a few early sampling-based ones can be found in Hwang and Ahuja's survey [20].

The work presented in this dissertation is substantially related to this branch of motion planning. The goal is to develop *global* solutions to the motion planning problem, which necessarily involves a consideration of the entire free space. Our approach to this is subdivide the free space using an appropriate cell decomposition: a convex polygonal decomposition in some cases, or a Collins decomposition

as used in the algorithm of Schwartz and Sharir. This implies that this work will have some of the same limitations as these motion planning algorithms. Two significant limitations are the difficulty of explicitly computing cylindrical algebraic decompositions, and the algorithmic complexity of these algorithms, which makes computing decompositions in high dimensions prohibitively expensive. As a result, we focus primarily on problems where polygonal decompositions can be used, thereby avoiding expensive algebraic decompositions (although theoretical results are presented for these as well). Many mobile robotics problems are relatively low-dimensional, so algorithms utilizing cell decompositions can be effectively applied to them.

Over time, motion planning research tackled increasingly difficult problems, in higher dimensional configuration spaces. For these problems, approaches based on cell decompositions were difficult or impossible to apply. As a result, researchers sought to solve motion planning problems without explicitly computing representations of the obstacle region. Instead, algorithms were developed that used collision detection algorithms to "probe" the configuration space to determine if a particular point (or, in some cases, region) is in $\mathcal{C}_{free}$ or $\mathcal{C}_{obst}$. These algorithms are call sampling-based motion planning algorithms, because of this feature. Early approaches of this type include [21–23]. Discretization and grid search were used by [24–26]; the planner by Kondo uses "lazy" sampling and collision detection [25], a philosophy followed by a number of later algorithms, such as lazy PRMs [27–29].

The first sampling-based motion planning algorithm to gain significant popularity was the Randomized Path Planner of Barraquand and Latombe [30]. This work influenced the trend toward randomized motion planning algorithms in the 1990s. Two major branches of this trend are probabilistic roadmaps (PRMs) [31] and Rapidly-exploring Random Trees (RRTs) [32, 33]; other notable methods

include Ariadne's Clew Algorithm [34, 35], Expansive Space Trees [36], PDST-Explore [37], and Probabilistic Cell Decompositions [38].

Probabilistic roadmaps sample the configuration space and attempt to connect nearby free samples with a collision-free path computed using a simple local planner (most often, a straight line path; however, other alternatives have been studied [39]). The original PRM, along with its numerous extensions and variants (e.g., [28, 40–48]), have been successfully applied to problems in robotics, computer animation, and computational biology [49–51].

Rapidly-exploring Random Trees explore the configuration space by iteratively building a search tree. In its basic form, a sample is drawn from the configuration space; then, its nearest neighbor in the tree is found. That node then expands in the direction of the sample, generating a new node in the tree. RRTs are applicable both to the original piano mover's problem and to the MPD problem, and have generated a significant number of applications and extensions [48,52–68].

While the bulk of motion planning research has been targeted toward systems without differential constraints, the consideration of problems with differential constraints is a growing field of research. Laumond introduced nonholonomic planning to the community [69]. Barraquand and Latombe gave the first resolution complete planner for nonholonomic multibody vehicles [70]. A number of approaches are based on decomposing the problem into two parts: first, planning a path for the system without considering the constaints; then, transforming the path to satisfy the constraints [56, 71–74]. Other methods attempt to directly compute an input trajectory which solve the problem [33, 37, 55, 75–77]. Most of these algorithms are probabilistically complete; [55, 75, 77] are resolution complete. Frazzoli et al. introduced the idea of maneuver-based motion planning for nonlinear systems with symmetries [59, 78].

1.2.3 Control theory and feedback

In contrast to motion planning, which focuses on finding a feasible (i.e., collision-free) open loop trajectory to a goal state from a given initial condition, control theory is concerned with the global problem of guaranteeing that the goal will be reached from any initial condition. Consider a system with state space $X = \mathbb{R}^n$, input space $U \subseteq \mathbb{R}^m$, and state transition equation $\dot{x} = f(x, u)$. One of the most important attributes that the system can possess is *stability*; two key kinds of stability are defined below.

Consider an uncontrolled system ($\dot{x} = f(x, 0) = f(x)$) and a corresponding equilibrium point $\bar{x}$ (a point where $f(\bar{x}) = 0$).

Definition 1.3 *An equilibrium point $\bar{x}$ is stable in the sense of Lyapunov if for every $\epsilon > 0$, there exists a $\delta > 0$ such that*

$$||x(0) - \bar{x}|| < \delta \Rightarrow ||x(t) - \bar{x}|| < \epsilon, \forall t \geq 0.$$

This means that in the neighborhood of the equilibrium point, all initial conditions have bounded system response. Note that this does not imply that the system has bounded response from any initial condition, although these are sometimes equivalent, as in the case of a linear system. A stronger type of stability is asymptotic stability:

Definition 1.4 *An equilibrium point is asymptotically stable if it is stable in the sense of Lyapunov and if there exists $\delta > 0$ such that*

$$||x(0) - \bar{x}|| < \delta \Rightarrow \lim_{t \to \infty} x(t) = \bar{x}.$$

If the state converges to the equilibrium point for any initial condition, the system is *globally asymptotically stable*. If an equilibrium point is not stable in the

sense of Lyapunov it is said to be *unstable*. Many uncontrolled systems are not stable; one of the central goals of control theory is to compute feedback controllers (alternatively, *feedback laws* or *control laws*) so that the controlled system is stable, even if the uncontrolled system is not. The definition of a feedback law is simple:

Definition 1.5 *A feedback law is a map* $\pi : X \to U$.

A *closed loop* system is one whose state evolves under the influence of a control law; in this situation, the state transition equation has the form $\dot{x} = f(x, \pi(x))$. The goal, then, is to design the control law so that the closed loop system has a stable equilibrium point. It is even more desirable for the equilibrium point to be asymptotically stable, and to be able to choose the location of that equilibrium point, through the design of the control law. Another common goal is for the closed loop system to have trajectories which are optimal according to some cost function.

1.3 Problem Formulation and Basic Approach

The problem addressed in this dissertation is formally charactereized in this section, and broad solution ideas will be introduced. The problem formulation must incorporate the requirements of both obstacle avoidance and global convergence, in order to adequately encapsulate the challenges facing contemporary robotics. Likewise, the utility of this approach depends not only on its formal correctness (which is nonetheless essential), but also on its efficiency, implementability, and ease of use.

1.3.1 Global navigation in the presence of obstacles

Control theory generally considers takes the state space to be free of obstacles; however, this is not realistic for robotics applications. Instead of freely designing control laws over the entire state space X, we are restricted to control laws that cause the robot to remain in X_{free}. This problem—computing globally asymptotically stable controllers that avoid obstacles—is the focus of this dissertation. This will be referred to as the the *global navigation problem*, which is defined as follows. Given a goal set x_g,

Definition 1.6 *A control law π solves the global navigation problem if it has the following properties:*

1. **Safety:** *If $x(0) \in X_{free}$, then $x(t) \in X_{free}$ for all $t > 0$.*

2. **Convergence:** *For all $x(0) \in X_{free}$, $\lim_{t\to\infty} x(t) \in x_g$.*

In some cases, it may be necessary to consider a relaxation of the convergence requirement above. The *semiglobal navigation problem* is as above, with the exception that there may exist equilibrium points outside of x_g, which have zero measure. There are two types of solutions to the semiglobal navigation problem. First, a control law *weakly* solves the semiglobal navigation problem if all equilibrium points outside of x_g have a region of attraction[2] of zero measure. Second, a control law *strongly* solves the semiglobal navigation problem if the region of attraction of any equilibrium point outside x_g is the point itself. For some of the problems considered in this work, the planning algorithms solve the global navigation problem; for others, only solutions to the semiglobal navigation problem will be achieved.

[2]The region of attraction of an equilibrium point is the set of points that asymptotically converge to the equilibrium point.

In addition to safety and convergence, a third desirable solution criterion is *smoothness*. A control law π is smooth if it is a C^∞ function of X. Similarly, a control law π smoothly solves the global navigation problem if all solution trajectories of $f(x, \pi(x))$ are C^∞ functions of time. In an environment with obstacles, it is generally not possible to achieve the former (smoothness of the control law over X), but the latter (trajectory smoothness in time) is often attainable. The term *smooth feedback plan* will be used to refer to a control law that smoothly solves the global or semiglobal navigation problem.

1.3.2 Cell decompositions and vector fields for smooth feedback planning

Consider a fully actuated robot with $\dot{x} = u$, in an n-dimensional state space $X = \mathbb{R}^n$. Without obstacles, the control law $u = -x$ globally stabilizes the robot to the origin; for any initial state x_0, applying the control law yields $x(t) = e^{-t}x_0$. Why is global stability so easy to achieve in this case? In addition to the trivial dynamics of the system, the lack of obstacles plays a critical role. The fact that the robot can always move directly toward the goal point, irrespective of whether there might be obstacles in the way, means that the stabilizing controller is easy to construct.

This fact directly leads to the first crucial element of our approach to smooth feedback planning, which is the use of cell decompositions to partition the configuration space (or state space) into simple regions. Since it is relatively easy to construct stabilizing controllers when there are no obstacles, it should also be easy to construct controllers over local obstacle free cells. The global navigation problem is then decoupled into discrete and continuous components. The environment is decomposed into a discrete set of cells together with a graph describing their connectivity; discrete algorithms can be applied to determine how to navigate

from any cell to the cell containing the goal point. Within each cell, a continuous problem must be solved: how to design a control law that causes the robot to move toward the next cell, from any point in the current cell. Since each cell is free from obstacles, the robot can always move from cell to cell without colliding with any of them.

The second key element of this approach addresses the problem of how to solve the continuous control problem within each cell. A simple approach might be to try the approach above, causing the robot to move straight at the face that separates it from the cell it wants to enter. This approach is globally stabilizing and safe; however, the trajectory of the robot is not smooth, and the control law is not even continuous. Developing a controller that is smooth—not only within each cell, but also across cell boundaries—is much more challenging. To achieve this, we define local smooth vector fields over the cells of the decomposition, and use a smooth interpolating function to cause the robot to follow one or more of these vector fields, smoothly transitioning between them as necessary.

This approach, which uses cell decompositions to partition the configuration space into simple pieces and local vector fields to construct a smooth control law over the decomposition, is efficient and useful for a large number of practical applications. A great many robotics applications can be modeled in two-dimensional environments, for which there exist many efficient cell decomposition algorithms. Finding vector fields to guide the robot from cell to cell is surprisingly simple, as we will see in Chapter 2. In contrast to many other approaches (which will be reviewed shortly), the smooth feedback planning algorithms presented here sacrifice neither formal completeness guarantees nor usefulness and efficiency in practice. As such, they make a valuable contribution in the area of global robot navigation and mobile robotics.

1.4 The Organization of This Dissertation

This chapter has presented the motivation for this work, provided some of the relevant historical background to this work, formally defined the problem to be addressed, and outlined the approach of this work and its significance. The next three chapters will these smooth feedback planning algorithms in detail.

Chapter 2 develops a solution to smooth feedback planning for robots without differential constraints (i.e., $\dot{x} = u$). Simple differential constraints will be considered in Chapters 3 and 4; initially, however, it will be more fruitful to narrow our focus to the core problem of developing control laws in obstacle cluttered environments, without considering dynamics. First, we describe smooth feedback planning in arbitrary-dimensional spaces with piecewise linear obstacles. Second, we present an algorithm for the general problem of smooth feedback planning in spaces in which the obstacles are semi-algebraic sets. This is an important theoretical result, because the classic piano mover's problem has exactly this formulation. Finally, approaches to smooth feedback planning for planar disc and polygonal robots in polygonal environments will be discussed.

Chapter 3 considers smooth feedback planning for simple nonholonomic systems. One such system is the unicycle, which models a vehicle with a single no-slip constraint (such as that arising from a wheel-ground contact). A unicycle is able to move forward and backward, and rotate in place; however, it is not permitted to translate in any direction other than the direction it is oriented in. The unicycle is used to model a differential drive vehicle, which is commonly used in robotics. In addition to the unicycle, we discuss smooth feedback planning for car-like vehicles. Car-like vehicles are limited by a path curvature constraint; they are unable to rotate in place as unicycles do. As a result, the smooth feedback plans developed for these vehicles will incorporate smooth "reversal" maneuvers

in which the robot alternates between moving forward and backing up, so that it can go around corners that the curvature constraint prevents it from traversing in a single motion.

Chapter 4 presents two extensions of this approach to smooth feedback planning. First, we will consider trajectory tracking controllers; given a smooth curve in the plane, we will see how to construct a smooth feedback controller that will cause the robot to converge to the curve while avoiding obstacles. Smooth feedback planning will also be applied to the problem of centralized multiple robot coordination. Chapter 5 will conclude and offer suggestions for future work.

CHAPTER 2

SMOOTH FEEDBACK FOR FULLY ACTUATED ROBOTS

2.1 Background and Motivation

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.

2.1.1 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 [79–82]. 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 [11]. Such algorithms have been extensively studied [8,83,84]. 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) [85] and Probabilistic Roadmaps (PRMs) [31]. If the robot is not actually kinematic and holonomic, then the paths produced by these algorithms need postprocessing to be transformed into feasible trajectories for the system; this is generally referred to as *decoupled trajectory planning*. Postprocessing methods include time-scaling [86, 87], steering [88, 89], or other transformations [56, 71–73]. 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) [33, 90], Expansive Space Trees (ESTs) [36, 76], and PDST-Explore [37]. Other approaches include the use of mixed integer programming, which computes optimal paths for problems with polygonal obstacle constraints and piecewise-affine system dynamics [91–93]. 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 replanned (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 conver-

gence to the goal state cannot be guaranteed, even though one might informally expect convergence to occur.

2.1.2 Closed loop navigation

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 nonzero 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 [94, 95]. 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 [96]. Many additional references for potential fields for robot navigation can be found in [97–99]. 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 [100–103]. 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 [104–106]. Wang and Chrikjian simulate steady state heat transfer in [107]. Waydo and Murray use stream functions for navigation in two-dimensional environments [108].

One of the most influential potential field techniques is that of Rimon and Koditschek [109]. 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 [110–115].

There are a number of other local navigation approaches based on potential fields. These include the Virtual Force Field (VFF) [116] and the Vector Field Histogram (VFH) [117] and their extensions, VFH$^+$ [118] and VFH* [119]. 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 [120, 121] and the Dynamic Window Approach [122] attempt to combine local and global navigation by choosing a control that is 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;

once again, convergence is not guaranteed. The dynamic window approach is extended in [123–125].

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 [126]. Bohlin used Green kernels to compose a workspace potential using samples from $SE(3)$ [127]. Elastic roadmaps append local potential fields to the nodes of probabilistic roadmaps to provide robustness in the face of dynamic environments [128].

One alternative to potential field methods is velocity field control. 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 parametrization. It was introduced by Li and Horowitz [129–131]; stability of the system is demonstrated using notions of passivity. Velocity field control has been applied frequently to robot manipulators [132–134], with the velocity field specified over the operational space of the manipulator. Velocity fields have also been applied to wheeled mobile robots [135–137].

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 [138–143]. 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.

2.1.3 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 [144–150]. One particular type of hybrid controller is based on sequential composition of funnels [151–153]. 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. The local controller must guarantee safety for the cell, as well as convergence to appropriate cell edge. Safety can be guaranteed by ensuring that the controller is inward pointing along the entire boundary of the cell; this is a common condition, similar in principle to the ideas behind polytopic Lyapunov functions [154–158]. If the controller for each cell safely funnels the robot to the exit 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 [159–162], rectangles [163, 164], or general polytopes [165, 166] (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 [167].

Conner et al. use local potential fields to define control policies on individual polygonal cells [168]. 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 [169].

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 [77, 170]. Our methods give stronger smoothness results than the above methods, and have been extended to a unicycle robot [171] and a car-like robot [172].

2.2 Point Robots in Piecewise Linear Environments

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 a convex polytope. 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 [173, 174]. The input to our algorithm is the cell complex and a goal state x_g. As discussed in Section 1.3, 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.

2.2.1 Description

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

1. Given a 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 Consider the d-cells and $(d-1)$-cells of the cell complex. Under the convention that the free state space 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 consider only these cells, we will henceforward 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). 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 2.1 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 obstacle avoidance and to guarantee adherance to the discrete plan; in other words, the face vector fields prevent the robot from crossing a cell face corresponding to an obstacle boundary or an improper cell transition, instead causing the robot to

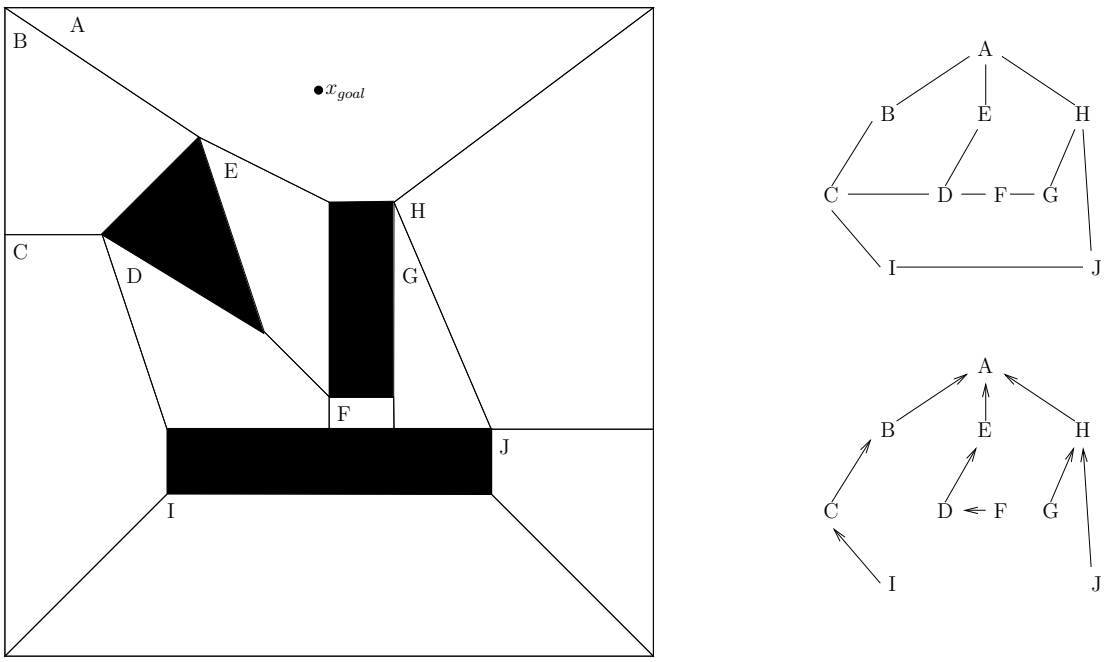

Figure 2.1: An environment decomposed into convex cells, and the corresponding connectivity graph and discrete plan.

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 2.2 for an illustration of face vector fields.

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

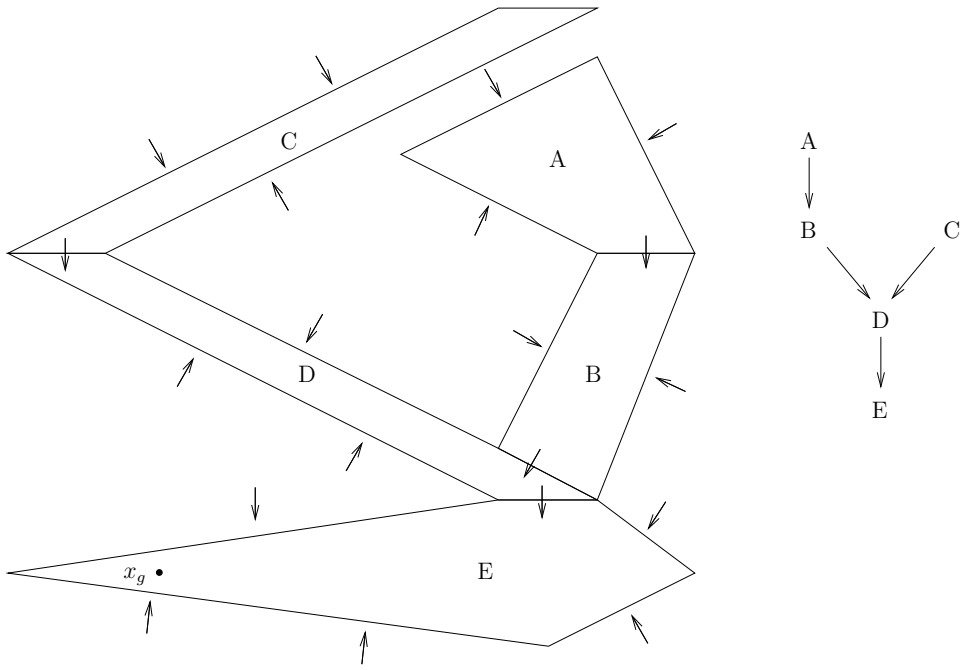

Figure 2.2: Vector fields assigned to faces of cells in the environment.

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 vector fields simultaneously. A natural choice for this is to use the generalized Voronoi diagram (GVD) of the cell [175, 176]. 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 a pair of faces. See Figure 2.3.

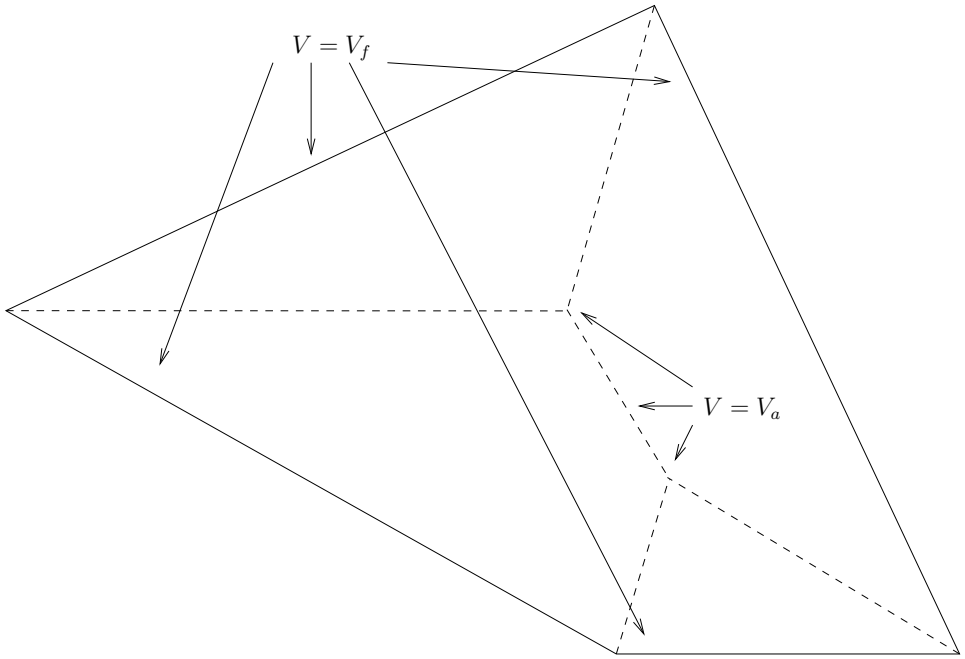

Figure 2.3: A cell, partioned using the generalized Voronoi diagram. The dashed lines are the GVD surface, and the shaded region is the region of influence (Voronoi region) of face f_i.

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*:

Definition 2.1 *Let X be a smooth manifold, and let K be a closed set and U an open set, $K \subset U \subseteq X$. A bump function over U is a smooth, real-valued function $\rho : X \to [0, 1]$ such that:*

1. *ρ has support contained in U.*

2. *$\rho(x) = 1$ for every $x \in K$.*

A bump function that transitions smoothly from 0 to 1 on the unit interval is constructed as follows. First, define

$$\lambda(s) = (1/s)e^{-1/s}. \tag{2.1}$$

The bump function is then defined as

$$b(s) = \begin{cases} 0 & s \leq 0 \\ \frac{\lambda(s)}{\lambda(s)+\lambda(1-s)} & 0 < s < 1. \\ 1 & 1 \leq s \end{cases} \tag{2.2}$$

An illustration of this bump function is given in Figure 2.4. The bump function has the important property that all derivatives equal zero at the endpoints of the unit interval.

Proposition 2.1 *For any i, $d^ib/ds^i(0) = d^ib/ds^i(1) = 0$.*

Proof: Clearly, $d^ib/ds^i(0^-) = d^ib/ds^i(1^+) = 0$; therefore, consider the derivatives in the open unit interval. From Equation (2.1), we see that

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

It is clear that all higher derivatives are likewise zero because the exponential factor dominates the polynomial factor. Equation (2.2) yields

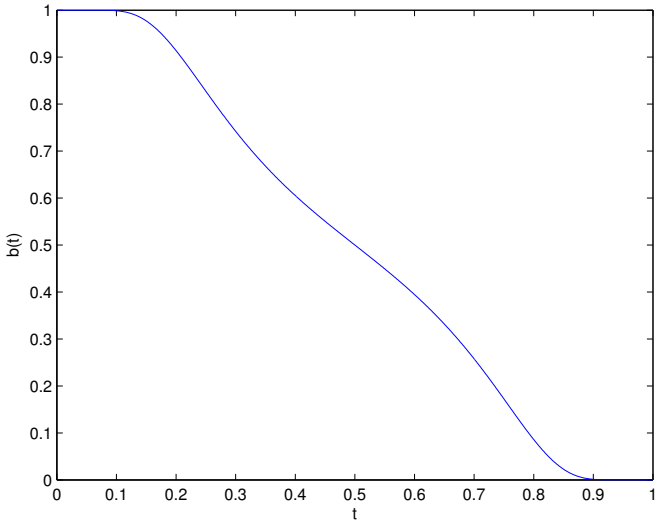

Figure 2.4: A bump function.

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

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, $\frac{db}{ds} \to 0$. It is clear that all higher derivatives go to zero as well. ■

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

$$s(p) = 1 - \prod_{j \neq i} \frac{d(p, f_j) - d(p, f_i)}{d(p, f_j)}. \tag{2.3}$$

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 $d(p, f_j) = d(p, f_i)$ is satisfied for some j. Therefore, the function 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

$$V(p) = \text{norm}\left(b(p)V_f(p) + (1 - b(p))V_c(p)\right), \tag{2.4}$$

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 norm 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.

2.2.2 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 2.2 *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 (2) 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 $V_c(x) = \text{norm}(p-x)$, in which $p \in \bar{C} \backslash C$ is fixed. A variety of similar constructions are possible. See Figure 2.5 for an illustration.

Proposition 2.2 *As defined above, V_c is a cell vector field.*

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

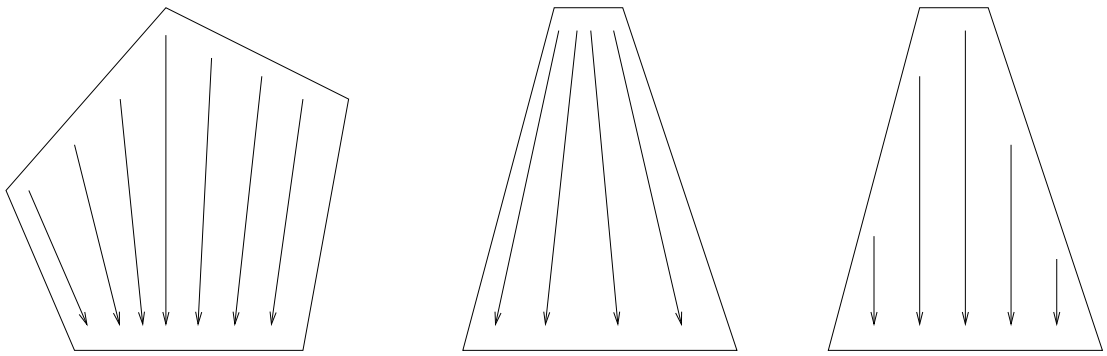

Figure 2.5: Three possible cell vector fields. Each points toward the exit face at the bottom of the cell.

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 2.3 *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 2.1 *Under Definitions 2.2 and 2.3 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 2.2. 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 2.2; 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 (nongoal) 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: $V_c(p) = b(||x_g - p||)\text{norm}(x_g - p))$ to satisfy this condition. This is smooth, satisfies the inner product requirement, and decays to zero at the goal point.

Theorem 2.2 *All integral curves in the goal cell C_g asymptotically converge to the goal point.*

Proof: This statement is proven similarly to Theorem 2.1. 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 2.3 *The integral curves of the vector field V, defined over X_{free}, asymptotically converge to the goal state x_g.*

Proof: From Theorem 2.1, 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 2.2 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 2.4 *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. ■

2.2.3 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 unkown, 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 [177]. 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: $O(n^{2^d})$ [178].

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 [174] 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 [179], and a practical implementation based on Seidel's algorithm is available, which requires $O(n \log^* n)$ time [180]. In practice, these algorithms can decompose even large and complicated environments in milliseconds, on modern desktop computers.

2.2.4 Discussion and computed examples

Before turning our attention to the case of feedback plans over cylindrical algebraic decompositions, we will discuss several practical issues associated with the use of our method. We will discuss the impact of choice of face and cell vector fields on path quality and the applicability of our approach to dynamic environments. We also give several computed examples, seen in Figures 2.6-2.8.

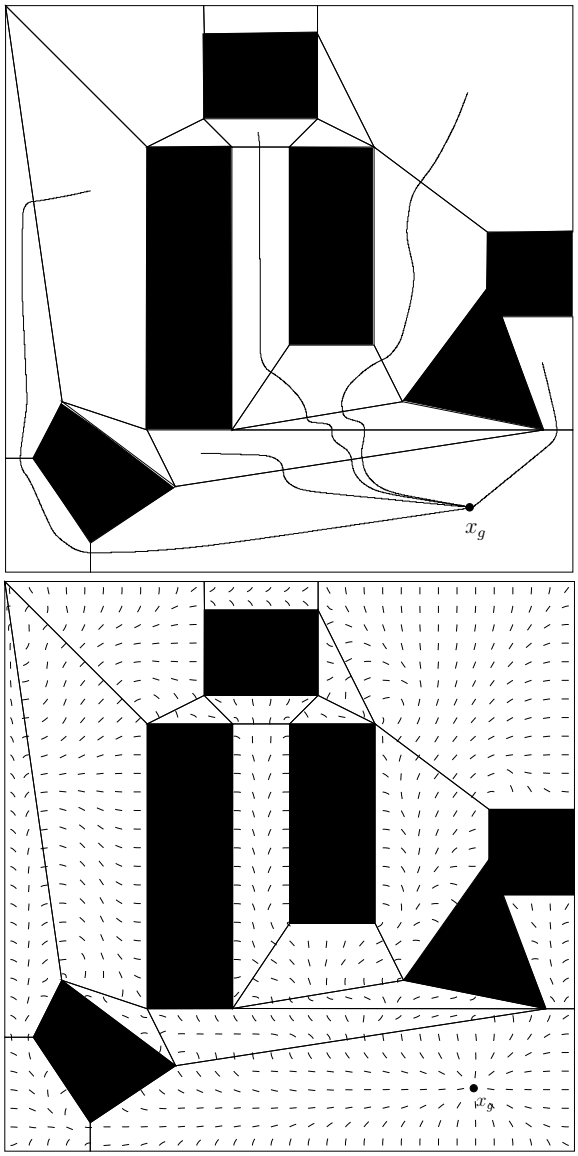

Figure 2.6: A computed example; the left figure shows several system trajectories, and the right figure illustrates the entire vector field.

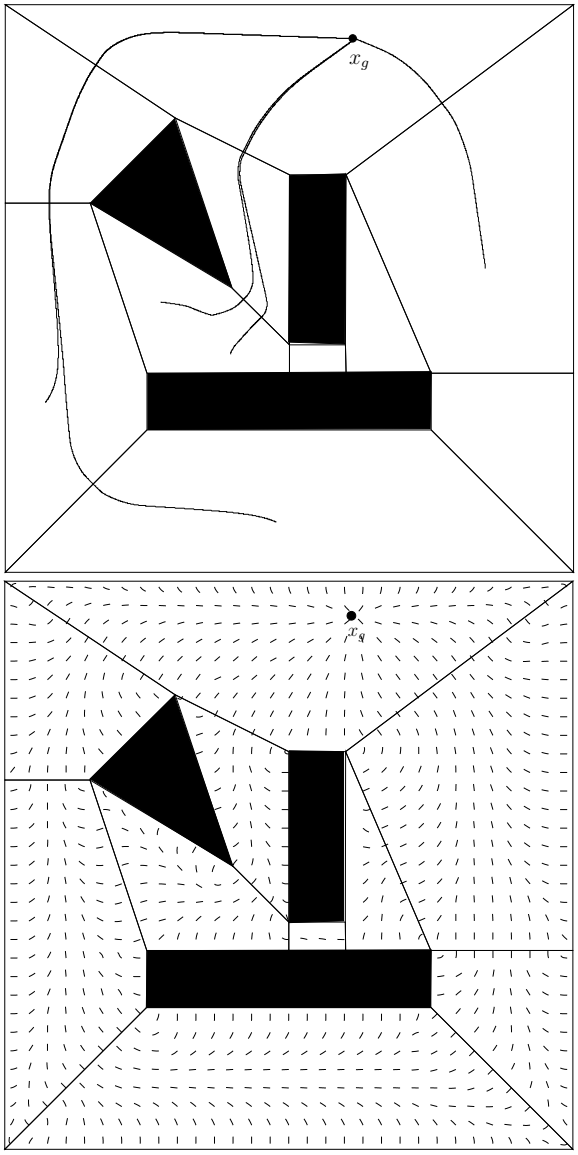

Figure 2.7: A second computed example.

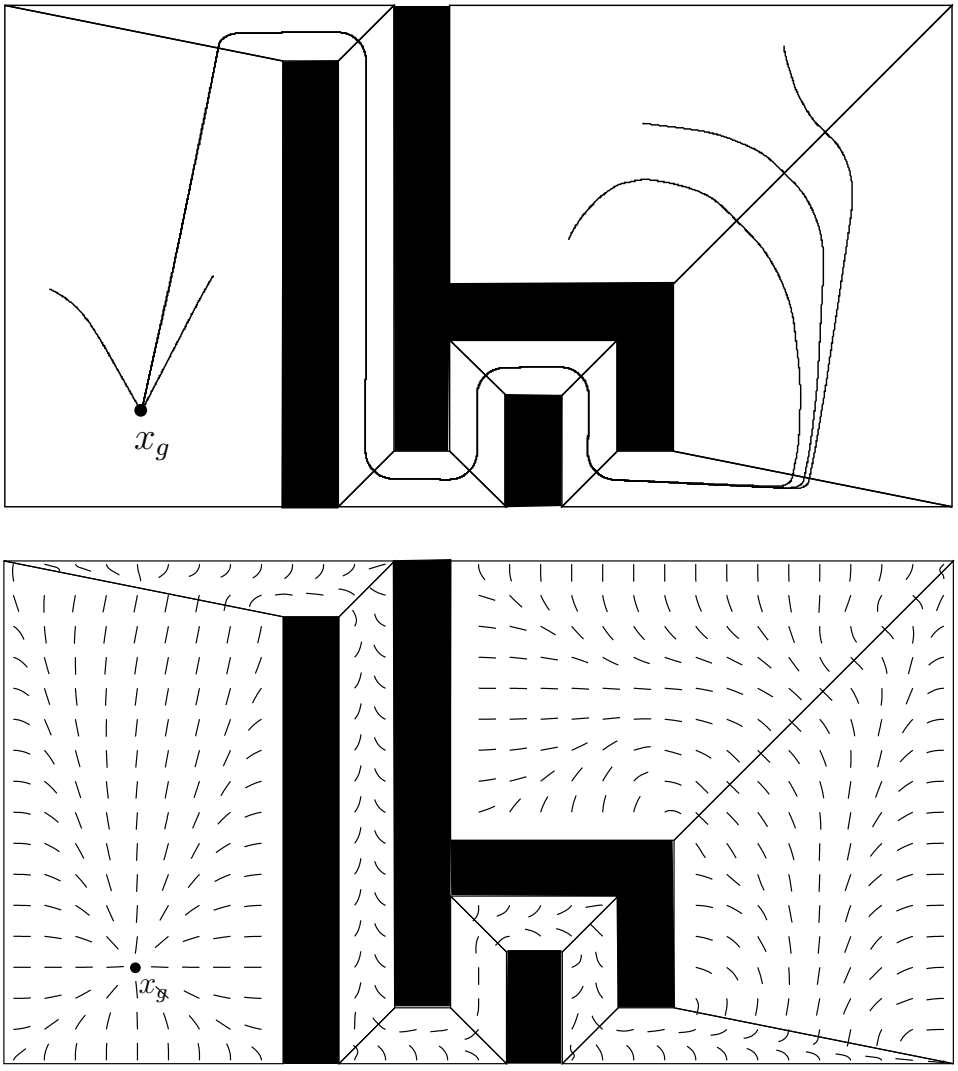

Figure 2.8: A third computed example.

Designing face and cell vector fields

In Section 2.2.2, we stated fairly general conditions on the face and cell vector fields under which convergence is guaranteed. The purpose of doing this is to permit a great deal of design flexibility, making our algorithm suitable for practical application. 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 which is equidistance 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. There still remains a great deal of design freedom 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 V_f that could be advantageous in terms of generating high-quality paths. First, we might want the vector field to point toward the exit face as much as possible, to promote short paths. Second, we might actually want it to point as far away from the exit face as possible, to avoid the sharp turns that the first approach might induce. Third, it might be preferable to simply make each face vector field perpendicular to its face, which is the simplest approach. Finally, we might wish to compute the

centroid of the cell or of the exit face and direct the vector field toward it (say, from the centroid of the face). In different situations, each of these approaches could offer advantages; this greatly depends on the particular application.

Any one of these approaches can be followed in the presence of the face vector field constraints. The constraints specified as sufficient for convergence are all simple inner product constraints (i.e., each constraint requires that the vector field have positive inner product with the normal of some hyperplane). As above, let f be a face under consideration, and denote the "bisector" hyperplane between f and f_x by b_f. Denote the unit normal vector of this hyperplane as n_{bf}. We require that the face vector field have positive dot product with n_{bf} (note that in the class of constant vector fields, every vector field with $V_f \cdot n_{bf} > 0$ satisfies $V_f \cdot n_{bf} \geq \epsilon$ for some $\epsilon > 0$). If the desired vector field does not satisfy this condition, simply project the vector field onto b_f, adding to it an arbitrarily small fraction of n_{bf} to attain a positive dot product. If the face vector field is also the exit face vector field of a previous cell, we have more constraints to apply. Their application, however, is identical to that just described. Finally, it is worth noting that *a face vector field aligned with the face normal always satisfies the constraints we have outlined*. If a different vector field is desired, each constraint must be examined and satisfied in order for convergence to be guaranteed.

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 $V_c(x) = \text{norm}(p-x)$. 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

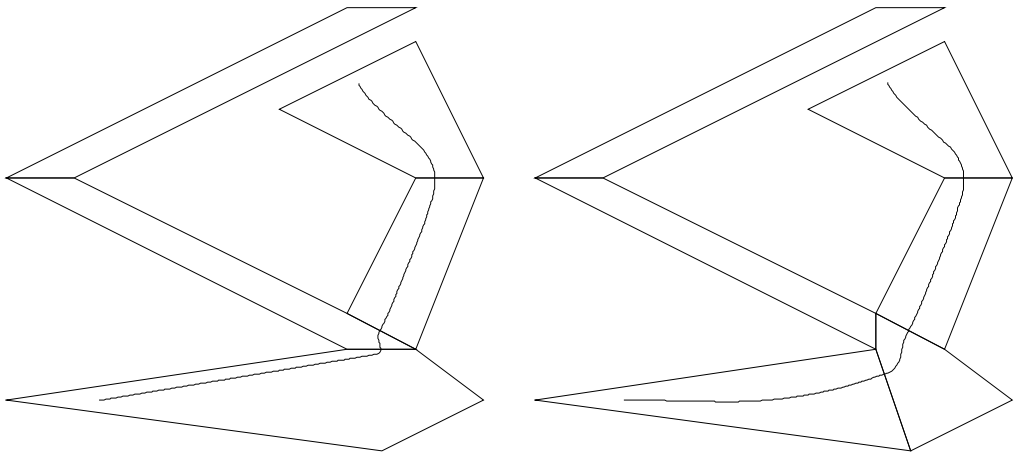

Figure 2.9: The influence of different convex decompositions. On top, a path with sharp turns arising from the choice of decomposition and perpendicular face vector fields; on bottom, the artifacts eliminated through a better decomposition.

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 2.9 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 obstacle face, 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 preform 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.

2.3 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. 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 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 [181] or a polygon translating and rotating in the plane [182]). In cases such as these, our algorithm has potential for practical implementation and use.

2.3.1 General feedback planning using cylindrical algebraic decompositions

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* [13], of $\mathbb{R}^n$ is defined in the following inductive way (see [15] for a more formal definition):

Definition 2.4

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 [13]. 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

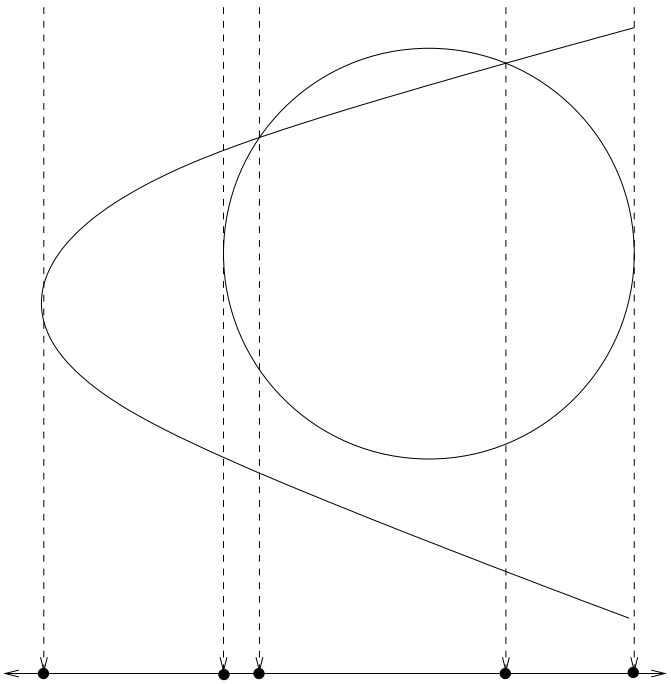

Figure 2.10: Two polynomials projected into $\mathbb{R}^1$, preserving the critical points and intersections. After the projection, each interval is lifted into $\mathbb{R}^2$ where it becomes a cylinder of cells. Each new 2-cell is sign-invariant under the polynomials.

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 [12, 15], 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 [8, 15, 183, 184]. A (very) simple illustration can be seen in Figure 2.10. Additionally, the algorithm can compute a single *algebraic point* in each cell of any dimension i, $1 \leq i \leq n$.

Schwartz and Sharir showed how to use the CAD algorithm to solve the generalized piano mover's 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 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 [12] for more details.

2.3.2 Algorithm assumptions

In order 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 [183, 185]), 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 [14, 15].

2.3.3 Algorithm description

We will now present our algorithm for generating smooth feedback plans over CADs. As in Section 2.2, we will construct smooth feedback over individual cells and then guarantee that smoothness is preserved across the boundaries crossed by the resulting integral curves.

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 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,

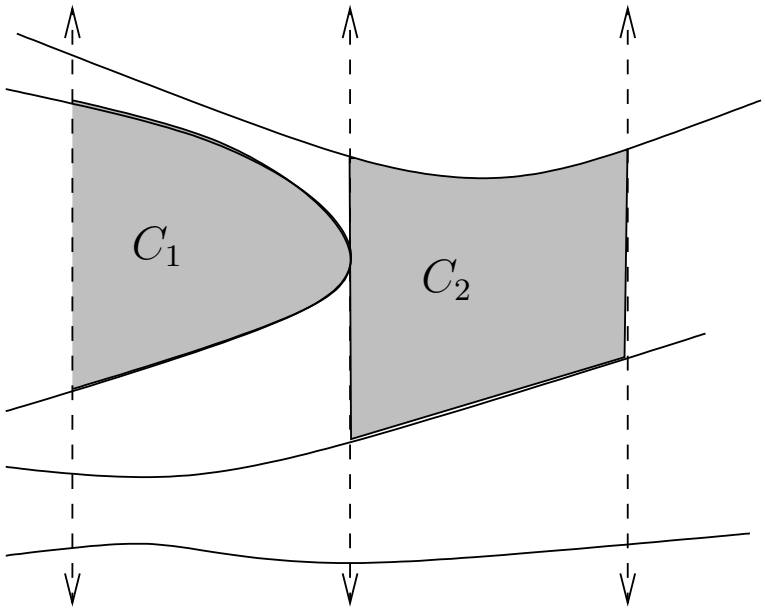

Figure 2.11: Adjacent cylinders in a cylindrical algebraic decomposition. Cell C_1 has no adjacent cells corresponding to one of its upper bounding polynomials, and cell C_2 has two adjacent cells corresponding to a lower bounding polynomial (the cells above and below C_1).

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 may seem 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 2.11. 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 2.12.

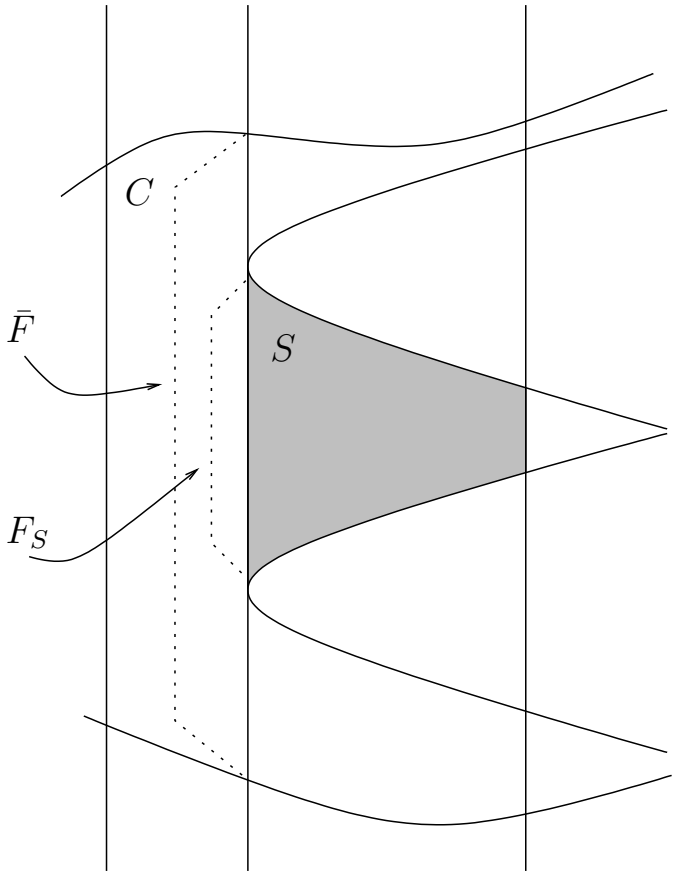

Figure 2.12: A cell C, its successor S, and the shared face F_S. The shared face is a hole in the larger face $\bar{F}$ of C. Lifting these cells into higher dimensions could continue to restrict the face they share.

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 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

$$h_i(x) = \frac{x_i - r(\mathcal{P}_{i-1}x, l)}{r(\mathcal{P}_{i-1}x, l+1) - r(\mathcal{P}_{i-1}x, l)}. \tag{2.5}$$

This is a smooth mapping, since both the upper and lower bounding polynomials are smooth (the root function maps onto the bounding polynomials). Even in the case where there are no actual faces corresponding to u_i and l_i, as in Figure 2.11, the bounding polynomials are well-defined. 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 2.11. 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 2.2, 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 $F_S \subseteq \bar{F}_i^{\pm}$. Then for any face $\bar{F}_j^{\pm}$, $j \neq i$, define the scaled perpendicular distance function $d_\perp$ as follows:

$$d_\perp(x, F) = \begin{cases} \frac{1-h_j(x)}{1-h_j(p_a)} & \text{if } F \subseteq \bar{F}_j^{+} \\ \frac{h_j(x)}{h_j(p_a)} & \text{if } F \subseteq \bar{F}_j^{-} \end{cases}. \tag{2.6}$$

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 $d_\perp(x, F) = |h_i(x) - h_i(p)|$. Another distance function will be defined, in order to distinguish F_S from the remainder of $\bar{F}_i^{\pm}$. This distance function will be used 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 2.12.

Keeping in mind that each successive lifted dimension adds constraints that may restrict the shared face between C and S, consider some point $x \in \bar{F}_i^{\pm}$. We can easily 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, 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 only depends 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 that 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

$$\begin{aligned} w_j(x) = & b\left(\frac{1}{\alpha}\left(\frac{h_j(x) - h_j(p_a)}{h_j(z_j^+(x)) - h_j(p_a)} - (1-\alpha)\right)\right) \\ & + b\left(\frac{1}{\alpha}\left(\frac{h_j(x) - h_j(p_a)}{h_j(z_j^-(x)) - h_j(p_a)} - (1-\alpha)\right)\right). \end{aligned} \tag{2.7}$$

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:

$$\begin{aligned} \hat{d}_{i+1}(x) &= w_{i+1}(x) \\ &\vdots \\ \hat{d}_j(x) &= \hat{d}_{j-1}(x) + (1 - \hat{d}_{j-1})w_j(x). \end{aligned} \tag{2.8}$$

The important results are summarized in the proposition below:

Proposition 2.3 *The following properties hold:*

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

Proof: We prove the first property by induction. As we have already indicated, $w_j(x)$ is only guaranteed to be well-defined if the polynomial constraints l_k^F and u_k^F are satisfied for all $k \in i+1, \ldots, j-1$. 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"). There are three different cases. Assume as before that $F_S \subseteq \bar{F}_i^{\pm}$. First, for some face

$\bar{F}_j^{\pm}$ with $j \neq i$, we say that x lies in the region of influence of $\bar{F}_j^{\pm}$ if $d(x, \bar{F}_j^{\pm}) \leq d(x, \bar{F}_k^{\pm})$, for all k. Second, we say that x lies in the region of influence of F_S if $d(x, \bar{F}_i^{\pm}) \leq d(x, \bar{F}_k^{\pm})$ for all k and if $\hat{d}_n(x) \leq 1 - \hat{d}_n(x)$. Finally, x lies in the region of influence of $\bar{F}_i^{\pm} \setminus F_S$ if $d(x, \bar{F}_i^{\pm}) \leq d(x, \bar{F}_k^{\pm})$ 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 2.2, we use a product of analytic switches. For any face $\bar{F}_j^{\pm}$ with $j \neq i$, use the following:

$$s(p) = 1 - \prod_{\bar{F} \neq \bar{F}_j^{\pm}} \frac{d_{\perp}(p, \bar{F}) - d_{\perp}(p, \bar{F}_j^{\pm})}{d_{\perp}(p, \bar{F})}, \tag{2.9}$$

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 $V(p) = \text{norm}(b(p)V_f(p) + (1 - b(p))V_c(p))$, 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 norm 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.

2.3.4 Theoretical results

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 2.2.

Theorem 2.5 *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 2.2). Hence all integral curves of V are smooth. ■

Theorem 2.6 *The integral curves of V remain in X_{free}.*

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

Theorem 2.7 *The integral curves of V asymptotically converge to the goal state.*

Proof: First, we show that for any nongoal 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, $h_j(x) \to h_j(p_a), 1 \leq j \leq n$. The face vector fields corresponding to all $\bar{F}_j^{\pm}$, $j \neq i$ also cause the relative coordinates to converge. The only exception is h_i, which 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. ■

2.4 Practical Feedback Planning in Planar Environments

Thus far, vector fields on two different types of cell decompositions have been described: convex piecewise linear decompositions and cylindrical algebraic decompositions. The former can be used to construct feedback plans over the exact configuration spaces of some robot models, as well as over piecewise linear *approximations* of other configuration spaces. The latter is used to construct feedback plans for the generalized piano mover's problem; it is extremely general, but has limited practical usefulness.

In this section, we consider the construction of feedback plans for planar disc and polygonal robots. The robots are fully actuated and move in a piecewise linear environment. First, we present an exact solution to the case of a disc robot; then, we show how to construct a conservative, arbitrarily accurate, piecewise linear approximation of a polygonal robot's configuration space which will enable the application of the results already described.

2.4.1 Disc robots

In general, planar robots with bodies have three-dimensional configuration spaces isomorphic to $SE(2)$; the configuration variables correspond to position and orientation and are usually denoted x, y, and θ. However, fully actuated disc robots have two-dimensional configuration spaces [8]. This is readily apparent, because the robot is symmetric about the center of its body.

Even for piecewise linear (PL) environments, the configuration space is not PL; instead, the configuration space boundary consists of both line segments and circular arcs. Algorithms for computing the configuration space obstacles are straightforward, and have been described in detail elsewhere [8, 84]; it is sufficient to note that configuration space obstacles are the Minkowski sum of the environ-

ment obstacles and the robot's body (i.e., a disc of some radius). Figure 2.13 illustrates a piecewise linear environment and the corresponding configuration space.

If the circular arcs are approximated by line segments, the approximate configuration space is polygonal and the algorithm described above can be directly applied. It is trivial to choose line segments so that the approximation is conservative, so strong safety guarantees can still be made in the approximate case. Since the arcs can be approximated to any resolution, this can be described as a *resolution complete* method as well. This will be discussed in more detail in Section 2.4.2 below.

While a polygonal approximation may be perfectly sufficient for most practical applications, it is interesting to explore whether or not smooth feedback plans can be easily constructed over an *exact* cell decomposition of these configuration spaces. The general method for cylindrical algebraic decompositions can be used, but this is probably not practical even for two-dimensional configuration spaces like these. Below, these configuration spaces will be decomposed into simple (though non-convex) cells, and vector fields such as those described above will be used to construct feedback plans that are sufficient to guarantee safety and convergence.

Decomposing the configuration space

As we have already seen, the boundary of the configuration space consists of line segments and circular arcs; assume this has already been computed. A simple algorithm to decompose the configuration space into simple is as follows:

1. Replace each circular arc with a line segment between its endpoints.

2. Perform a convex decomposition the resulting polygonal configuration space.

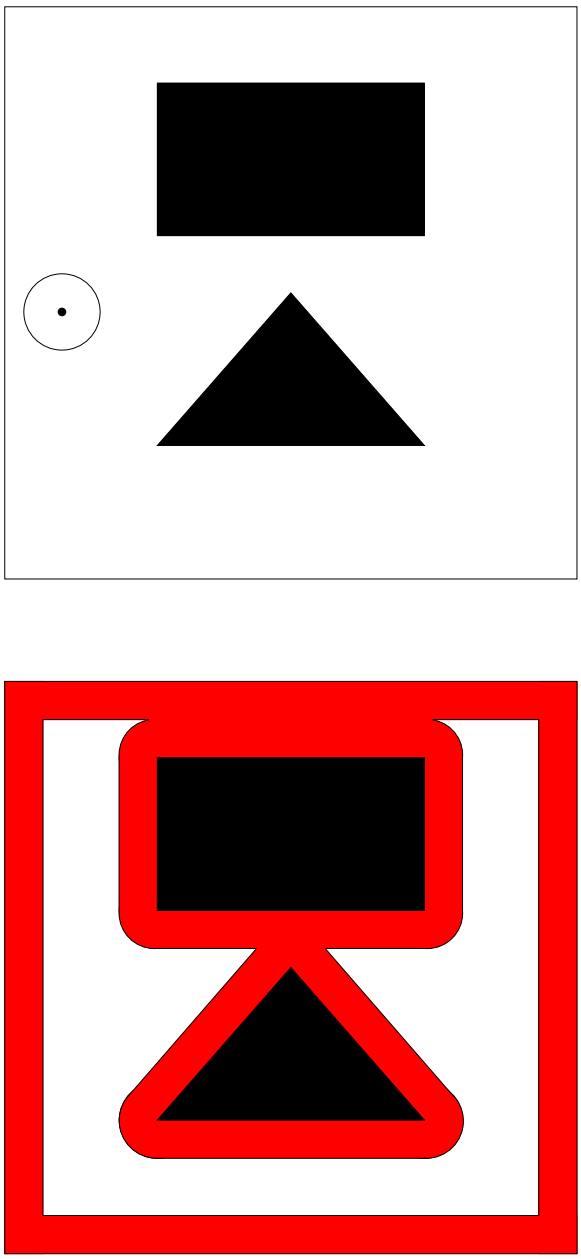

Figure 2.13: A piecewise linear environment, and the corresponding configuration space for a disc robot. The red area is part of the C-obstacle region, in addition to the original obstacles.

3. Reinsert the circular arcs as "edges" in the decomposition, subdividing arcs as necessary to respect the cells in the decomposition.

The first and second steps are easily completed, by any of the convex decomposition methods mentioned above. Therefore, we will consider the third step in more detail.

At the outset, note that the third step is always feasible, and that all of the resulting cells in the decomposition have edges which are line segments or circular arcs. It is simple to envision a convex decomposition superimposed on the true configuration space and see that this is true. As a result, it is important to specify only how the third step is carried out.

The insertion of the circular arcs causes the free configuration space to shrink, as the area inside the arcs is marked as in collision. However, observe that no inserted arc will enclose a vertex in the decomposition (or, as a result, an entire polygon in the decomposition). For this to be true, the decomposition must not include Steiner points, which are vertices in the decomposition that are not vertices of polygons in the environment. Most decomposition algorithms do not yield decompositions with Steiner points, so this is an insignificant restriction. The proof of this fact is simple: if the vertex is inside the arc, then it is not a C-obstacle vertex at all—it is in the interior of $\mathcal{C}_{obst}$, not on its boundary. The fact that no cells or vertices in the decomposition lie inside the inserted arcs is important because it greatly simplifies the way arc insertion can modify the cells in the decomposition.

Assume that we have a set of circular arcs to insert into the cell decomposition, and consider a polygon P whose edges intersect with a particular arc A. For convenience, make the general position assumption that no edges of P are tangent to A; also, adopt the convention that any edge that shares a vertex with

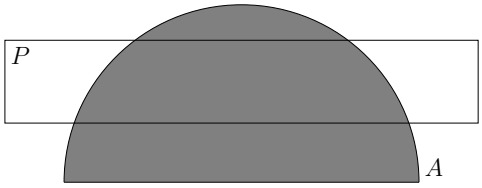

Figure 2.14: A portion of a polygonal cell decomposition, and an inserted arc. The arc intersects two edges of the polygon twice, dividing it into multiple cells.

A intersects A. If the edges of P intersect with A, then they do so in one of the two following ways.

Case 1: Two or more edges of P each intersect A twice. This implies that inserting A may separate P into multiple cells. See Figure 2.14 for an illustration. Computing the new cells is achieved by "walking" around the boundary of the polygon, dividing at the intersections.

Case 2: Exactly one edge of P intersects A twice. In this case, no other edge intersects A. No double intersection exists by the definition of this case, and no single intersection exists because that would imply the existence of a vertex inside the obstacle. Therefore, P is not subdivided; the edge which intersects A is divided into three new edges, two of which are line segments and one corresponding to A. This is illustrated in Figure 2.15.

These are the only two possible cases, because any other would imply the existence of a vertex inside the arc. For example, there cannot be a case as appears in Figure 2.16, in which the arc intersects once each with two edges. In each of the two possible cases, it is simple to modify the polygon so that the inserted arc A is taken into account. Doing so clearly requires linear time in the edges of the decomposition and the number of arcs to be inserted.

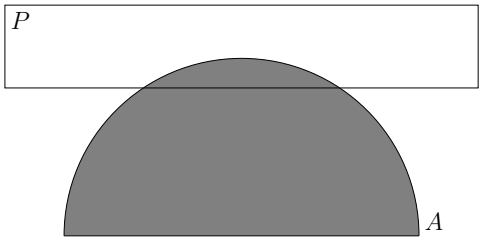

Figure 2.15: Part of a polygonal cell decomposition, and an inserted arc. The arc intersects a single edge of the polygon twice, which does not divide it into multiple cells.

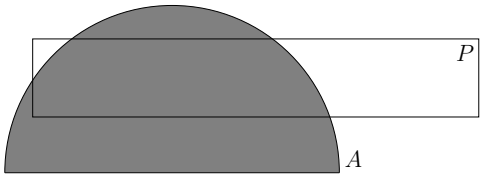

Figure 2.16: An arc cannot intersect a polygon edge only once, because that implies the existence of a vertex inside the obstacle.

Selecting local vector fields

In Section 2.2.2, we saw conditions for face and cell vector fields over convex PL cells, and the interpolation between them. These conditions guarantee that every integral curve in an intermediate cell will reach the exit face of that cell while avoiding other faces. Now, we will see how to choose local vector fields for these cell decompositions (bounded by straight line segments and arcs) so that the same guarantees can be made.

First, consider the cell vector fields. In Section 2.2.2, the restrictions on V_c were made to be as loose as possible. For the sake of simplicity, assume here that V_c is a constant vector field such that $V_c \cdot n_x > 0$, in which n_x is the outward pointing normal of the exit face. For face vector fields, the necessary modifications are also straightforward. In the original case of convex polygonal cells, we required (for nongoal cells) that $V_f(p) \cdot n > 0$ and $V_f(p) \cdot n_{bf} > 0$ for any face f other than the exit face, in which n is the face normal and n_{bf} is the normal of the

hyperplane equidistant from f and the exit face. Adjusting the first requirement slightly, require that for any point p on an arc edge, $V_f(p) \cdot n(p) > 0$, in which $n(p)$ is the arc's normal vector at p. It is easy to show that using a constant vector field perpendicular to the chord connecting the endpoints of the arc satisfies this condition. Note that every arc in the decomposition has an angular length less than π. This is because each arc is generated by the robot as it sweeps around a vertex of a polygon in the environment. Assuming that there are no degenerate polygons in the environment (having a vertex with an interior angle of 0) implies that the arc in configuration space must have angular length less than π. Then the following theorem applies:

Theorem 2.8 *Consider a circular arc A with radius r, and angular length $l < \pi$. Then $n_c \cdot n(p) > 0$ for any $p \in A$, in which n_c is the vector normal to the chord connecting the endpoints of A, and $n(p)$ is the outward pointing normal vector to the arc at point p.*

Proof: Consider an arc A' with angular length $l' = \pi$, and let the center of the circle containing A' be the origin. Then $n(p) = (\cos(\theta) \sin(\theta))^T$, for $\theta = arctan^{-1} p_y/p_x$ for $p \in A'$. We also have $n_c = (01)^T$. Then $n_c \cdot n(p) = n_c^T n(p) = \sin(\theta)$, which is strictly greater than zero for $\theta \in (0, \pi)$, which corresponds to the interior of A'. Since $A \subset A'$, this implies that $n_c \cdot n(p) > 0$ holds for all $p \in A$. ■

As a result, it is equally easy to select a feasible face vector field in this case as it is in the polygonal case. An requirement for the goal cell is identical.

Theoretical results

As in Section 2.2.2, we will establish that all integral curves of the vector field V, obtained by interpolating between the cell and face vector fields, are safe,

asymptotically converge to the goal state, and are smooth. The proofs are similar to those above.

Theorem 2.9 *In any nongoal cell, all integral curves of V reach the exit face in finite time and do not contact any other face.*

Proof: First, it is clear that no integral curve in V will contact any face other than the exit face. The vector field V is identically equal to the face vector field V_f on any face, and we have already seen that all face vector fields are strictly inward pointing, with the exception of the exit face. Therefore, we simply need to establish that V always makes sufficient progress toward the exit face, so that it will reach it in finite time. This is simple to establish from examining the inner product of V with n_x, the normal vector of the exit face. It is clear that $V \cdot n_x > 0$, because $V_c \cdot n_x > 0$ over the entire cell, and $V_f \cdot n_x \geq 0$ over the entire cell. As a result, the integral curve is always making progress toward the exit face; this is lower bounded away from zero by the fact that the face vector field for the exit face, V_{f_x}, satisfies $V_{f_x} \cdot n_x > 0$, so the vector field V is outward pointing on the exit face. Therefore the integral curves of V reach the exit face in finite time. ■

Theorem 2.10 *In the goal cell, every integral curve of V asymptotically converges to the goal state without reaching any cell face.*

Proof: Again, no cell face is reached, because all face vector fields are strictly inward-pointing for the goal cell, and $V = V_f$ on any face. Therefore, we only need to establish that the integral curves of V asymptotically converge to the goal state. Since the cell vector field V_c always points at the goal point x_g, and the face vector fields always satisfy $V_f(p) \cdot (x_g - p) > 0$ for any point p in the goal cell, it is clear that the interpolated vector field must also satisfy $V \cdot (x_g - p) > 0$ for any p. Then the distance to the goal point x_g is always decreasing; a strictly positive

lower bound is easily established as above, resulting in asymptotic convergence to x_g. ■

Theorem 2.11 *The integral curves of V are smooth.*

Proof: As we have seen throughout, the face and cell vector fields are smooth, and the the vector field V is generated by smoothly interpolating between them. Therefore V is smooth. ■

2.4.2 Polygonal robots

Unlike disc robots, polygonal robots that translate and rotate in the plane have a three-dimensional configuration space. Even though both the obstacles and the robot body are polygons, the configuration space obstacles have nonlinear boundaries. While cylindrical decomposition can be used, this section describes an approximation-based method that is extremely efficient and provides safety, convergence, and smoothness guarantees. The heart of the idea is to compute a polygonal approximation of the configuration space, thereby enabling the use of the algorithm developed in Section 2.2. The approximation proposed here is *conservative*, so strong safety guarantees can still be made.

It is well known that for any fixed orientation, the two-dimensional cross section of the configuration space is polygonal [8, 84]. The *star algorithm*, proposed by Lozano-Pérez [10], is an algorithm for computing the C-space obstacle boundaries for any such cross section. He also proposes creating polygonal approximations of the C-space slices corresponding to a range of orientations, using swept volumes. Lengyel et al. propose discretizing the configuration space and using graphics hardware to plan paths in real time [26]; this also results in a polygonal approximation of the three dimensional configuration space.

The star algorithm for convex robots and obstacles

The algorithm used for constructing conservative polygonal approximations of C-space slices is based on the star algorithm [10], which computes the configuration space obstacle corresponding to a robot with a convex body translating in an environment with a single convex obstacle. We briefly describe the star algorithm and then develop the approximation method.

The key to the star algorithm is recognizing that the C-obstacle region can be obtained by sliding the robot along the perimeter of the obstacle (again, we only permit the robot to translate for the time being). The resulting C-obstacle is also a convex polygon, the edges of which are generated either by sliding a vertex of the robot body along an edge of the obstacle, or sliding an edge of the robot body along a vertex of the obstacle. These are typically referred to as Type VE (vertex-edge) and Type EV (edge-vertex) contacts, respectively. The edges and vertices whose contacts generate the C-obstacle edges are easily determined by placing the edges of the robot and obstacle in an ordered list, according to the angles of their normal vectors. The normal vectors of edges of the robot body are taken to be inward pointing, and those of the obstacle edges are taken to be outward pointing. The list has topology S^1, with the beginning of the list connected to its end. See Figure 2.17 for an example.

To generate the C-obstacle boundary, traverse this list in order. Each edge will generate a C-obstacle edge. If the edge is from the robot, the vertex it contacts will be the vertex shared by the nearest obstacle edges on either side of it in the list; if from the obstacle, the vertex will be the one shared by the nearest robot edges on either side of it in the list. The C-obstacle edge will be the edge traced by the origin of the robot's body frame as the edge slides along the vertex. Figure 2.18 continues the example from Figure 2.17.

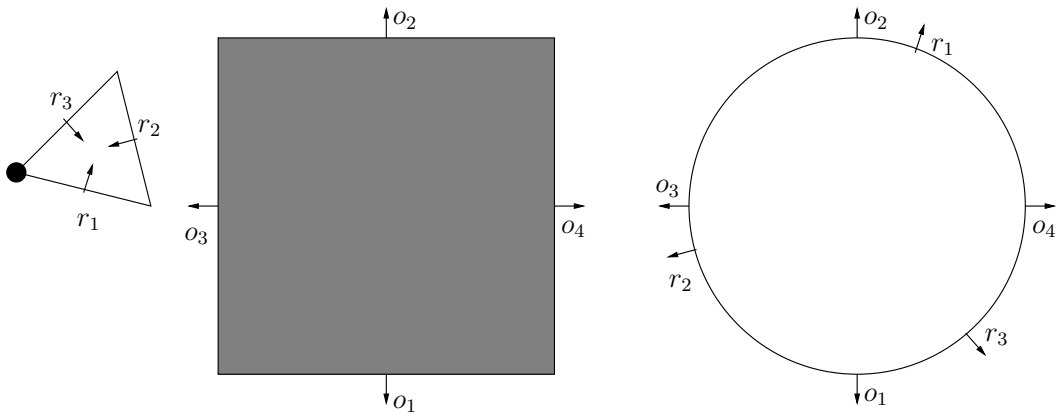

Figure 2.17: A triangular robot and a square obstacle; their normal vectors, sorted in a circular list.

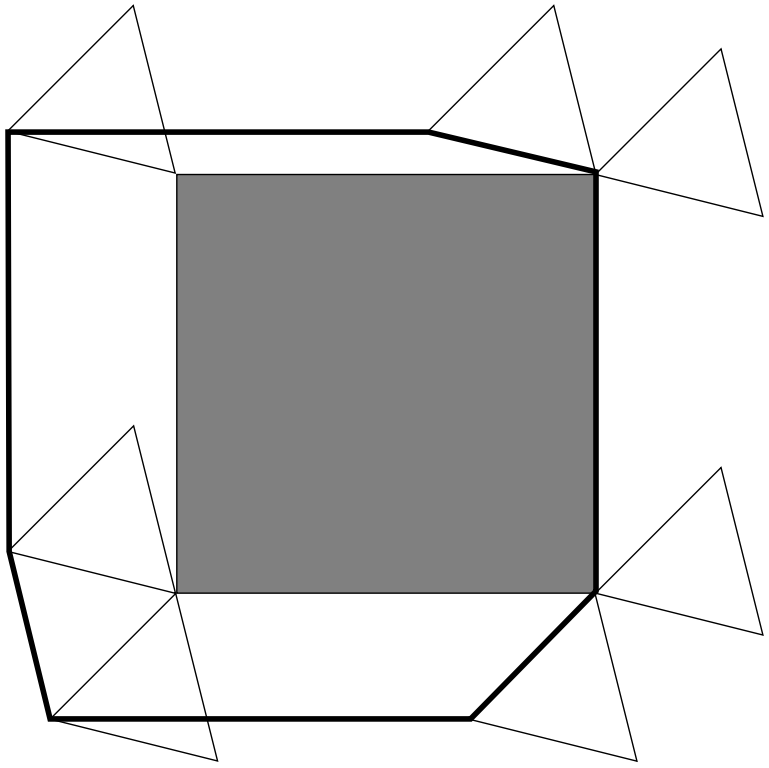

Figure 2.18: The configuration space obstacle generated by sliding the robot around the environment obstacle. The outline is the origin of the robot (as seen in Figure 2.17).

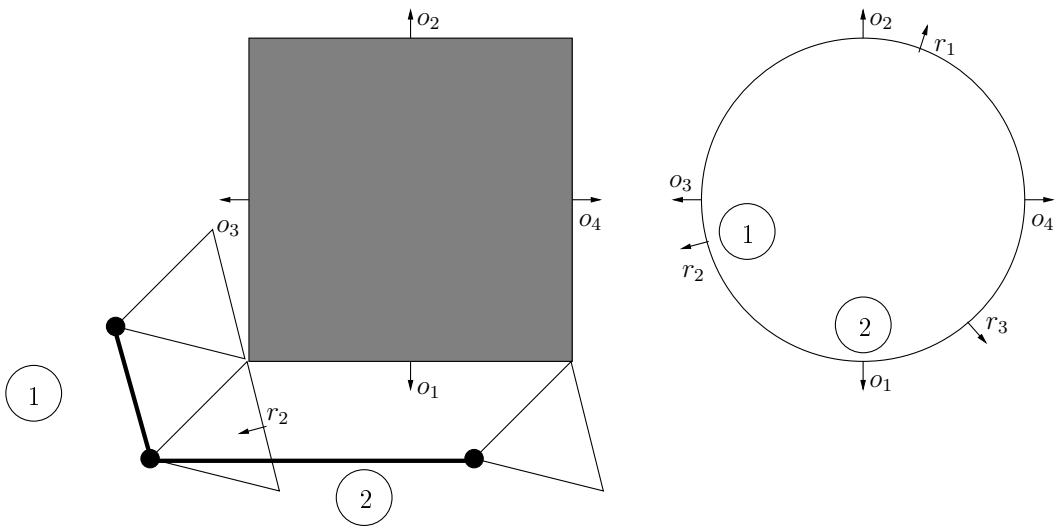

Figure 2.19: Two edges in the C-obstacle, generated by a EV contact and a VE contact, respectively.

Simple geometry is sufficient to derive the equations describing the edges of the resulting C-obstacle. First, consider the case of an edge generated by a Type VE contact. Let the position of the robot vertex be (x_v, y_v) in the robot body frame, and the edge vertices be (x_{e1}, y_{e1}) and (x_{e2}, y_{e2}) in the world frame. For a cross section with $\theta = \theta_0$, the position of the vertex is

$$\begin{pmatrix} x_{vn} \\ y_{vn} \end{pmatrix} = \begin{pmatrix} \cos\theta_0 & -\sin\theta_0 \\ \sin\theta_0 & \cos\theta_0 \end{pmatrix} \begin{pmatrix} x_v \\ y_v \end{pmatrix} = \begin{pmatrix} x_v\cos\theta_0 - y_v\sin\theta_0 \\ x_v\sin\theta_0 + y_v\cos\theta_0 \end{pmatrix}.$$

As the robot slides from (x_{e1}, y_{e1}) to (x_{e2}, y_{e2}), a C-obstacle edge is generated with endpoints $(x_{e1} - x_{vn}, y_{e1} - y_{vn})$ and $(x_{e2} - x_{vn}, y_{e2} - y_{vn})$.

The case is similar with a Type EV contact. Let the edge of the robot body have vertices with positions (x_{r1}, y_{r1}) and (x_{r2}, y_{r2}), and let the obstacle vertex have position (x_o, y_o). Under the same transformation as above, we have new robot vertex positions and the generated C-obstacle edge has endpoints $(x_o - x_{r1n}, y_o - y_{r1n})$ and $(x_o - x_{r2n}, y_o - y_{r2n})$. Both cases are illustrated in Figure 2.19.

Approximating C-space slices

The first step in creating a polygonal slice approximation method is to determine the slices themselves. That is, we must choose the orientations $\{\theta\}_1^N$, $0 \leq \theta_1 < \cdots < \theta_N < 2\pi$, so that each slice $[\theta_i, \theta_{i+1})$ is easy to approximate (note that $[\theta_N, \theta_1)$ is also a slice). From the discussion above, it is apparent that as the robot rotates, there are certain critical points where the circular ordering of edge normals changes; this occurs when the normal vector of a robot edge aligns with the normal vector of an obstacle edge. These critical points naturally partition the space into slices in which the C-obstacle has the same basic structure: namely, the C-obstacle boundary in each slice is generated by the same set of Type VE and Type EV contacts. Under a general position assumption, there are nm critical points (and nm corresponding slices) for a robot body with n edges and an obstacle with m edges.

With this slice structure, we need to determine a conservative polygonal approximation to the C-obstacle for each slice. How to do this becomes apparent when we examine how C-obstacle edges vary for different cross sections within the slice. First, consider a C-obstacle edge generated by a Type VE contact. The edges generated for any cross section within that slice are parallel, since they are all parallel to the obstacle edge; see Figure 2.20 for an illustration. Therefore, we need only to determine the angle for which the robot origin (and, therefore, the generated C-obstacle edge) is most distant from the obstacle edge. This is a simple maximization problem, which can be written as

$$\theta = \underset{[\theta_i, \theta_{i+1})}{\operatorname{argmax}} \left(\begin{pmatrix} (x_{e1} - x_{vn}) & (y_{e1} - y_{vn}) \end{pmatrix}^T n_e \right),$$

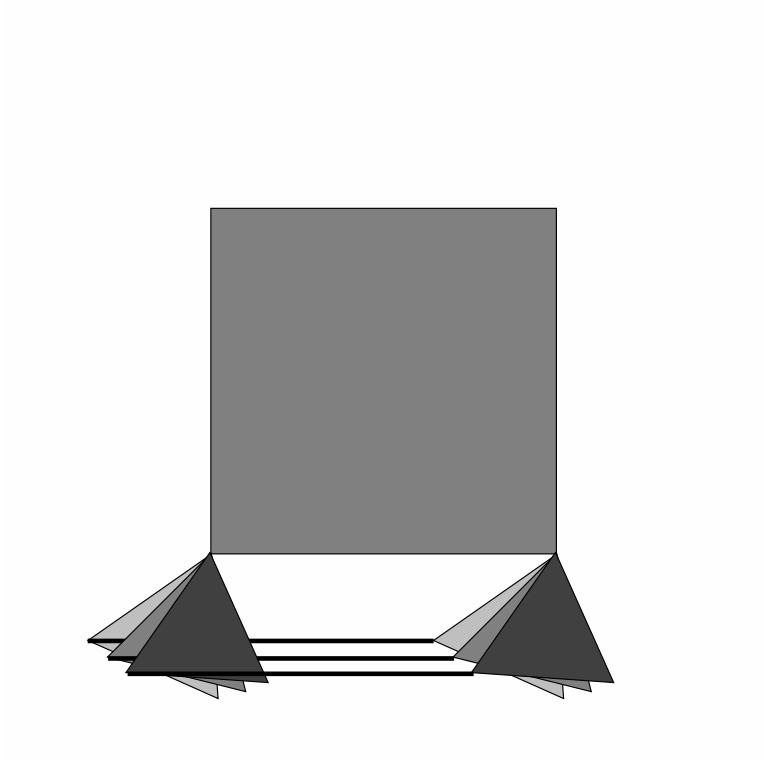

Figure 2.20: Several obstacle edges corresponding to different robot orientations within a slice, for a VE contact. The conservative approximation is the edge maximally distant from the environment obstacle edge.

in which n_e is the outward normal of the obstacle edge, and the other variables are as defined above. This is simple to solve; the conservative approximation for this C-obstacle boundary is the edge generated by the contact at this angle.

Having determined a conservative approximation of edges generated by type VE contacts, we must now construct an approximation of the edges generated by Type EV contacts (in which edges of the robot slide across obstacle vertices). There may be any number of Type EV contacts occuring in between two Type VE contacts; an illustration of two successive Type EV edges is given in Figures 2.21 and 2.22 (edges corresponding to two different angular orientations are shown). Successive Type EV edges will be referred to as a *chain*.

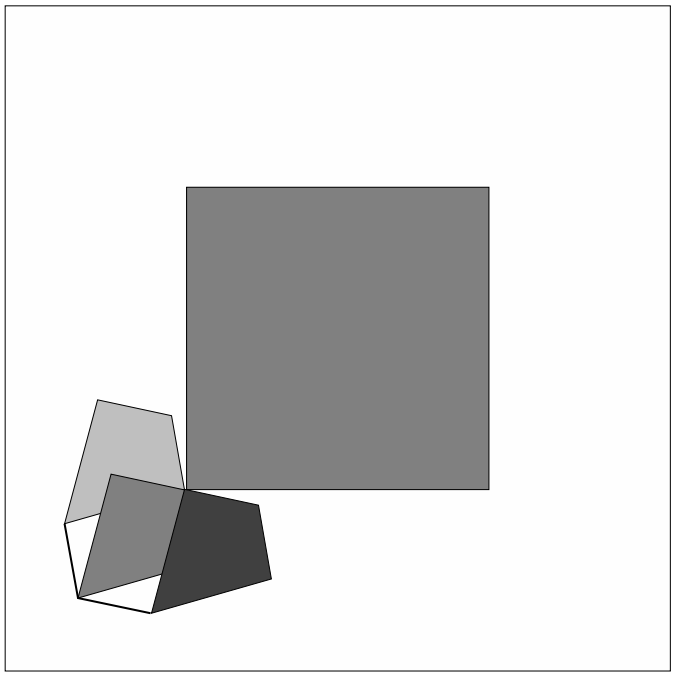

Figure 2.21: A chain of two successive Type EV edges.

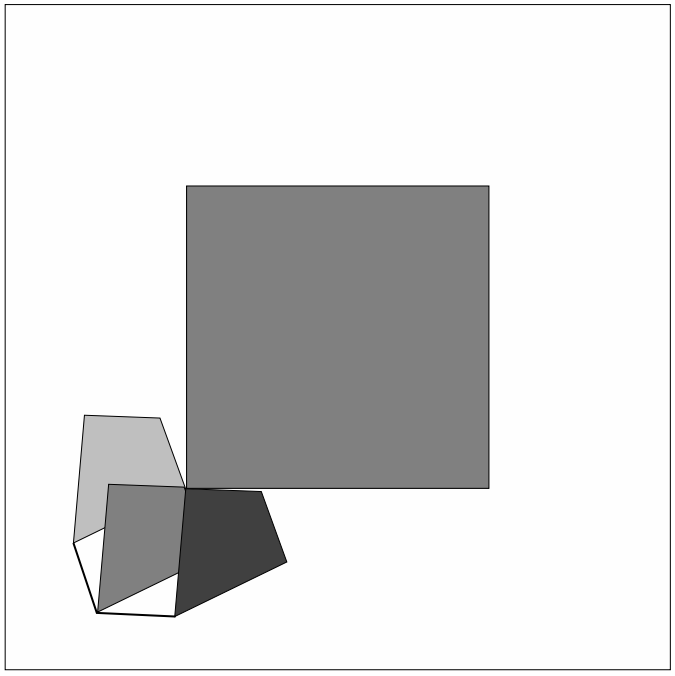

Figure 2.22: The same chain as shown in Figure 2.21, for a different angular orientation.

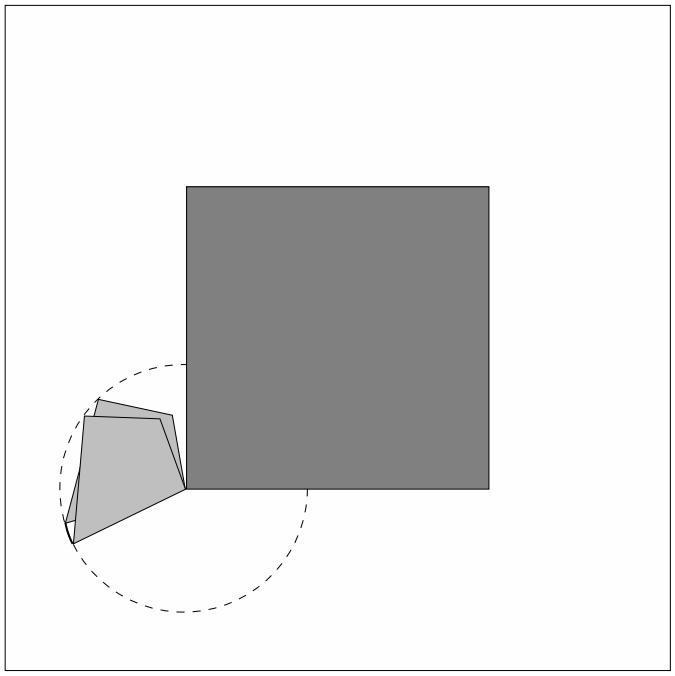

Figure 2.23: The motion of the robot origin as the robot rotates, for a first vertex in the chain.

Illustrations will clarify this discussion; consider the motion of the robot origin as the robot rotates between the angular orientations seen in Figures 2.21 and 2.22. Since the chain is of length two, there are three vertices in the chain (the endpoints of the chain are included). Figures 2.23-2.25 illustrate the motion of the robot origin over the angular interval, for each vertex in the chain. It is clear that the origin moves in a circular arc about the obstacle vertex, at a radius determined by the distance from the origin to the contact vertex.

Now, we need simply to create a conservative approximation for the volume swept as the chain rotates. To do this, we perform two steps. First, connect adjacent arcs with straight line segments such that the arcs are entirely contained in a same halfspace of the segments. Second, approximate the arcs themselves with straight line segments, as necessary. Consider the following lemma:

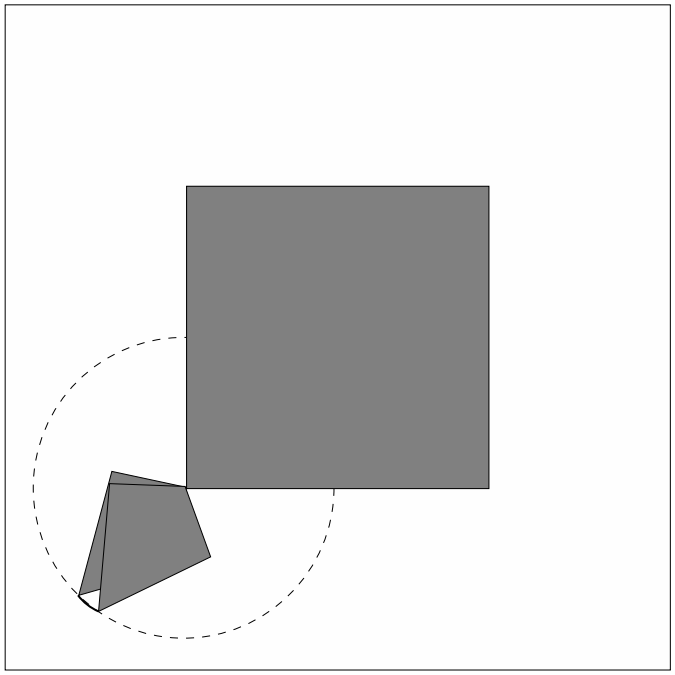

Figure 2.24: The motion of the robot origin as the robot rotates, for a second vertex in the chain.

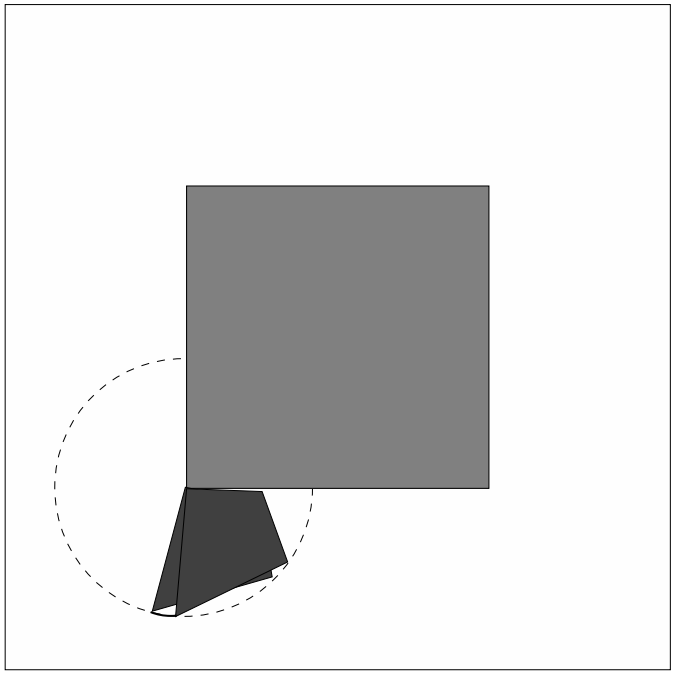

Figure 2.25: The motion of the robot origin as the robot rotates, for a final vertex in the chain.

Lemma 2.1 *Consider two circular arcs with different radii and a common center. From either endpoint of the arc of larger radius, there exist two line segments which are tangent to a circle with the smaller of the two arcs' radii. One of these contains both arcs in the same halfspace.*

Proof: Consider an endpoint on the arc of larger radius and a line tangent to the arc at that point. The arc is in a single halfspace of that line. Rotating the line in one direction about the endpoint will cause it to intersect with the arc, violating this condition. Rotate the line in the opposite direction. The line will then become tangent to the circle of smaller radius before it rotates far enough to intersect with the larger arc. This line segment (from the endpoint of the larger arc to the point of tangency to the smaller circle) contains each arc in the same halfspace. ■

This lemma indicates how to choose a line segment to connect two neighboring arcs: the segment should have one endpoint which is an endpoint of the arc of larger radius (the endpoint nearest to the other arc), and should be tangent to the circle with the smaller of the two arcs' radii. If the point of tangency is not part of the arc, the second endpoint may be shifted to the nearest endpoint of the arc. In either case, the entirety of both arcs will be on the same halfspace of the line segment, which is necessary for the conservative approximation. See Figure 2.26 for an illustration.

Having connected neighboring arcs with line segments, we simply need to approximate each arc itself. This is necessary if and only if there is a gap between the endpoints of the segments connecting it to its neighbors. It is obvious that this can be computed to any desired resolution, so this will not be described further.

Two important theorems need to be proved for this method: first, that the approximation is conservative; and second, that as the slice width goes to zero,

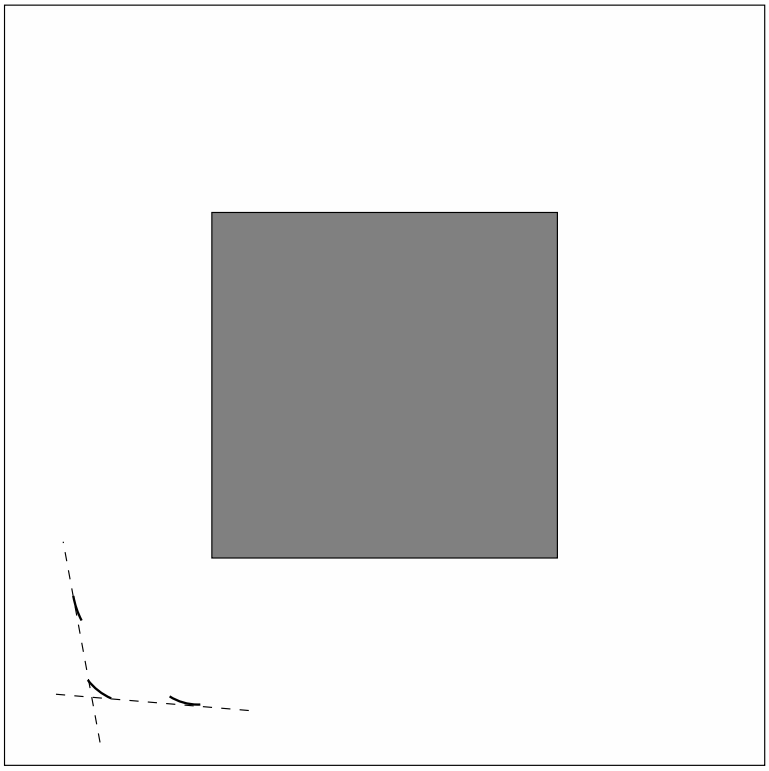

Figure 2.26: A conservative approximation is obtained by finding line segments that contain subsequent arcs in the same halfspace.

the approximation converges to the true C-obstacle cross section. This will imply that the approximation is resolution complete.

Theorem 2.12 *The polygonal slice approximation described above is a conservative approximation of the C-obstacle region.*

Proof: For every VE contact, the approximation is conservative because the corresponding edge in the cell is chosen to be the one that is the farthest from the obstacle edge; therefore, the entire obstacle region of the slice is in a single halfspace of the edge. For each chain of EV contacts, there is a corresponding set of arcs, corresponding to the movement of the robot origin as it pivots about the corresponding obstacle vertex, for one of the robot vertices in the chain. We described above how to connect subsequent arcs with a line segment such that both arcs lie in a single halfspace; this likewise implies that the entire obstacle region for the slice lies on that same halfspace. Therefore, the intersection of halfspaces for this sequence of edges is a conservative approximation. Since we have found conservative edges for each VE and EV contact, we have a conservative approximation of the entire C-obstacle region in the slice. ■

Theorem 2.13 *The polygonal slice approximation converges to the true C-obstacle cross section as the slice width goes to zero.*

Proof: In the case of VE contacts, the edge in the approximation clearly converges to the actual obstacle edge, as the argument of the maximum shrinks to a single point. Consider EV contacts. In the limit as as the width of the slice goes to zero, the length of the arcs in the chain also go to zero. Therefore, the line segments connecting them converge to the segments connecting the endpoints of the arcs; the endpoints of each arc converge, as the arc collapses to a single point. These endpoints are exactly the C-obstacle vertices. Therefore the slice

approximation converges to the true C-obstacle cross section as the slice width goes to zero. ■

Complex environments

The star algorithm, and the polygonal slice approximation algorithm based on it, are for a convex robot and a single convex obstacle. To handle more complicated scenarios, both robot and obstacles need to be decomposed into convex pieces, and the approximation algorithm performed for each resulting pair of robot/obstacle pieces. The polygonal approximations can be performed independently. After computing the approximations, the resulting polygonal C-obstacles are unioned together to form a global, polygonal (piecewise linear) $\mathcal{C}_{obst}$. The resulting piecewise linear configuration space can be decomposed and a smooth feedback plan computed, as described earlier in this chapter. Hence, we obtain a smooth feedback plan for a polygonal robot translating and rotating in a polygonal environment. Since the approximation can be made arbitrarily good, this algorithm is resolution complete.

One disadvantage of performing all of the polygonal approximations separately is that the slice structure is lost when the approximations are combined. (Recall that each robot/obstacle pair will have different slices, since the normal vectors of the obstacle edges will be different.) As a result, convex decomposition must be performed for a three-dimensional environment, while each slice is effectively only two-dimensional (since it is constant with respect to θ). The algorithms for convex decomposition in three dimensions are much more limited than the wide assortment of algorithms for planar environments. Therefore, one practical choice would be to force all polygonal approximations to have the same slices; when they are combined, the slice structure will remain and each slice can be decomposed

using a fast 2D algorithm.[1] Enforcing the constraint that each approximation have the same slices requires only that the union of the critical points for all approximations must be used to determine the slices for each one.

One minor problem with this approach is that because the slices each have different cross sections (and completely independent convex decompositions), the faces of cells will not match across cell boundaries. This is similar to what we saw in Section 2.3, in which cells might share only a partial face. Fortunately, the solution presented in that section works here as well. For example, assume that the exit face f_x of a cell is enclosed in the interior of a larger cell face f. Then, the face vector field should be inward pointing on $f \setminus f_x$, and smoothly blend to inward pointing over some subset of f_x. Any cell in a given slice with an exit face in the θ direction should be given an artificial "goal point" that projects into the interior of f_x. In this way, face vector fields can be chosen for the faces of the cell corresponding to the sides of the cylinder (i.e., the sides in the two-dimensional decomposition), and a vertical vector field (i.e., in the θ direction) will then suffice to guide the robot from one slice to another.

A more significant practical ramification of this approach is that the number of slices required to appxoimate a complex environment is greatly increased. In contrast, approximating the configuration space with a grid of fixed resolution is not dependent on the geometric complexity of the robot or obstacles. However, *accurately* approximating a complex configuration space requires a very fine grid resolution; hence, the decoupling of approximation size from geometric complexity is more illusory than actual in the case of the grid approximation. In comparison with the vertical decomposition algorithm, we have already argued that this ap-

[1] One possible algorithm to use in three dimensions is vertical decomposition; this algorithm chooses one dimension, partitions the space into slices along that axis, and recurses on each slice. Hence, it essentially does what we already implicitly did—sliced the configuration space along the θ axis.

proach should be comparable, because vertical decomposition would replicate the slice structure, at essentially the same "resolution."

2.5 Conclusion

In this chapter a family of algorithms for smooth motion planning have been presented. These algorithms are based on cell decompositions and vector fields; by combining local vector fields through smooth interpolating functions, we obtained feedback controllers that are globally asymptotically stable, safe, and smooth. Many previous approaches to feedback planning in obstacle cluttered environments rely on potential fields to define a control law. In these cases, one must typically choose between practical efficiency and guaranteed performance—complete algorithms are difficult to implement in practice, and efficient approaches suffer from problems with local minima in the potential function. In contrast, our approach is both complete and efficient for many practical robotics problems.

We first outlined the basic algorithm for a d-dimensional point robot moving in a piecewise linear environment. Next, the algorithm was extended to the feedback version of the generalized piano mover's problem, in which the robot and obstacles are semi-algebraic sets. As one of the most general formulations of the motion planning problem, this is extremely important from a theoretical perspective. The results presented here demonstrate that obtaining feedback solutions to this problem are no more difficult than obtaining an open loop path; hence, feedback comes essentially "for free," given the complexity of constructing a cylindrical algebraic decomposition. Finally, we presented practical solutions to the smooth feedback planning problem for planar robots with disc or polygon bodies. For disc robots, an exact decomposition of the configuration space was presented, yielding a complete solution to the smooth feedback planning problem. For polygonal

robots, a conservative polygonal slice approximation approach was used, enabling the algorithm for piecewise linear environments to be applied. In this case, the algorithm is resolution complete, though not exact.

These results are significant because these algorithms address the classic goals of control theory (global asymptotic stability) and motion planning (obstacles avoidance) in a way that is intuitive, theoretically rigorous, and efficient. In order to be fully relevant to practical robotics, it is important to address more complicated robot dynamics models. In the next chapter, we consider robots with simple nonholonomic constraints that are commonly seen in robotics.

CHAPTER 3

SMOOTH FEEDBACK FOR NONHOLONOMIC ROBOTS

3.1 Introduction

In Chapter 2, we presented a family of algorithms that address the smooth feedback planning problem. They construct global asymptotically stable controllers that have guaranteed obstacle avoidance and smoothness properties. In addition, they are simple to implement and efficient in practice. As such, they have much to offer robotics, from a practical as well as a theoretical perspective.

Thus far, we have considered only fully actuated robots, whose state transition equations are a trivial $\dot{x} = f(x, u) = u$. This greatly simplified the initial presentation of the planning algorithm, but it hardly addresses the needs of practical robotics. In contrast to free-flying robots, the motion of most mobile robots must satisfy certain nonholonomic constraints that restrict how they can move. For example, differential drive (or unicycle) robots can drive forward and rotate in place, but cannot slide sideways. Car-like robots, another important class of mobile robots, are even more restricted in their motion. They have a limited steering angle which prevents them from rotating in place; instead, their paths are restricted to having bounded curvature. This makes navigating around obstacles and through narrow passages extremely challenging.

3.2 Background and Motivation

Stabilization of nonholonomic systems in obstacle-free environments has been studied in depth [186–188]. Important research goals include stabilization to a point or trajectory and path following; an approach for computing optimal trajectories is presented in [189]. Unicycles are a simple nonholonomic system that have been studied in depth; in addition to unicycles, car-like robots are common and have received significant attention [89, 190–193].

When the environments are complex, the problem of global feedback control is especially difficult. Motion planning problems in robotics typically involve non-convex constraints resulting from obstacles in the environment, which present a significant problem for traditional feedback control methods. The standard way to solve problems with obstacles is to plan an open loop path using a motion planning algorithm, and then use the local controller to track the path or move from waypoint to waypoint. This approach can be reasonably successful in practice, assuming that the distance to the obstacles is large relative to the intra-waypoint distance. A severe weakness of this approach, however, is that it *cannot* generally guarantee both obstacle avoidance and global convergence; this is a result of decoupling planning and feedback. The algorithms developed in this chapter can make stronger guarantees by integrating the two more closely together.

As mentioned in Chapter 2, one may also use state space sampling together with dynamic programming to achieve not only feedback, but approximately optimal trajectories [138, 139, 143]. The curse of dimensionality limits the usefulness of this approach for any but low-dimensional spaces, however. Instead of solving the entire optimal control problem, one might try *receding horizon control*, in which an optimal solution is computed for a short time horizon, and new optimal solutions computed for each successive time interval [194–196]. Receding horizon

controllers have been developed for a number of interesting dynamical systems; one of the key challenges for this method (beyond the basic control problem) is the practical problem of computing optimal solutions fast enough to keep up with a tight control loop. Obstacle avoidance has been incorporated into receding horizon controllers through the use of mixed integer linear programming; see [93, 197] and the references therein.

Other approaches to feedback navigation in the presence of obstacles typically rely on scalar potential functions [94,109]. However, constructing such functions is quite difficult, and it is difficult to apply potential function or navigation function methods to systems with nonholonomic constraints (see [114,198] for an approach for robots with unicycle dynamics).

As in Chapter 2, the approach in this chapter is based on partitioning the state space into simple cells and constructing local controllers by smoothly interpolating between vector fields defined over individual cells. Several other approaches based on cell decompositions include [159, 162, 166, 168]. These generally consider the problem of control of an affine system on a arbitrary dimensional simplex or polytope.

3.3 Point Unicycle Robots

Consider the problem of a nonholonomic point robot navigating in a polygonal environment. We will use the kinematic unicycle model, the motion of which must satisfy the equation

$$\dot{x}\sin\theta - \dot{y}\cos\theta = 0. \tag{3.1}$$

The corresponding state space for this nonholonomic model is

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v_f \cos\theta \\ v_f \sin\theta \\ v_\theta \end{pmatrix}. \tag{3.2}$$

In addition to being a simple example of a system with nonholonomic constraints, this is a very natural model for the ubiquitous differential drive robot. The environment is a two-dimensional piecewise linear environment, $\mathcal{E}$, which is a bounded open subset of $\mathbb{R}^2$. The configuration space $\mathcal{C}$ is three-dimensional, consisting of position and orientation; formally, $\mathcal{C} = \mathcal{E} \times S^1$, in which $S^1 = [0, 2\pi]/\sim$, with $\sim$ an equivalence relation with $0 \sim 2\pi$. The *goal point* is defined to be the point $x_g \in \mathcal{E}$. The goal region, then, is $x_g \times S^1$. In other words, we ignore the orientation of the robot and consider only its position, for the purpose of solving the navigation problem. This is entirely reasonable; for the system we consider, Brockett's condition [199] implies that no static, smooth vector field (feedback control) can stabilize the complete state of the system. Additionally, it is often sufficient from a practical point of view to stabilize only the position of the robot. Once in the neighborhood of the goal state x_g, the orientation may be stabilized independently. For practical problems, it is also possible to automatically modify a cell decomposition so that the orientation is "mostly" stabilized when the robot approaches the goal point; this will be discussed briefly below.

3.3.1 Constructing smooth feedback plans

In this section, smooth feedback plans that satisfy the nonholonomic motion constraints while taking the robot to the goal region from any point in the environment will be discussed. The first step is to construct a smooth feedback plan over

$\mathcal{E}$ as discussed in Chapter 2. This is then used to construct a smooth feedback plan over $\mathcal{C}$.

A smooth feedback plan satisfying nonholonomic constraints

Assume that a smooth vector field V has been constructed over $\mathcal{E}$, as described in Chapter 2. In standard coordinates, for any $p = (x, y) \in \mathcal{E}$, we have

$$V(p) = v_x \frac{\partial}{\partial x} + v_y \frac{\partial}{\partial y}.$$

We use this to define the map $\theta_t : \mathrm{T}\mathcal{E} \to [-\pi, \pi)$ as $\theta_t(V(p)) = \tan^{-1}(v_y, v_x)$, in which $\tan^{-1}$ returns the four-quadrant arctangent of v_y and v_x. We can now define the *target manifold*:

Definition 3.1 *The target manifold $M_t \subset \mathcal{C}$ for a smooth vector field $V \subset \mathrm{T}\mathcal{E}$ is the set of configurations such that the orientation θ for each point matches the value of V at that point. Formally,*

$$M_t = \{(p, \theta) \mid \theta = \theta_t(V(p)), p \in \mathcal{E}\}. \tag{3.3}$$

Note that from any point on the target manifold, the nonholonomic system can follow the integral curves to the goal. Consider a time-parametrized integral curve $c(t)$. Recall that by definition, the tangent vector $T(t) = \frac{d}{dt}c(t) = V(c(t))$. Then the angular velocity of an oriented particle following the path is simply $\frac{d}{dt}\theta_t(T(t)) = \frac{d}{dt}\theta(V(c(t)))$, which is smooth since the vector field, integral curve, and function θ_t are smooth. The tangent space can be parametrized in terms of forward velocity v_f and angular velocity v_θ; to follow this integral curve with the nonholonomic system, simply set $v_f = 1$ (since the curve is parametrized with unit speed) and $v_\theta = \frac{d}{dt}\theta(V(c(t)))$. For $\theta_V = \tan^{-1}(v_y/v_x)$, this yields

$v_\theta = V(\theta_V)(q) = V(q) \cdot \nabla\theta_V(q)$. As a result, we can write an equivalent V as a vector field on $\mathcal{C}$:

$$V(p) = v_x(p)\frac{\partial}{\partial x} + v_y(p)\frac{\partial}{\partial y} + v_\theta(p)\frac{\partial}{\partial \theta}.$$

If the robot is not on the target manifold, then the vector field V is clearly not admissible (i.e., it does not satisfy the nonholonomic constraints); however, if we design an admissible vector field such that the target manifold is attractive and such that it equals the original V on the target manifold, then we will be able to conclude that all integral curves reach the goal.

It is well known that a simple controller can be designed that will cause the system to converge to the target manifold. For example, one can use a controller of the form $\dot{\theta} = \dot{\theta}_t - K(\theta - \theta_t)$ for some gain K, in which $\dot{\theta}_t$ is the derivative of θ_t while moving at the current velocity and heading. The most significant problem with this approach is that there are no guarantees that the obstacles will be avoided as the robot converges to the target manifold. In fact, for *any* predefined gain K, one can find configurations (positions and orientations) from which the controller would cause the robot to hit an obstacle. See Figure 3.1 for an illustration. In contrast, the method presented here guarantees safety by constructing a vector field over the configuration space which points away from the obstacles at the obstacle boundaries (or, at least, tangent to the obstacle boundaries). This is accomplished by smoothly interpolating between a nominal vector field V_n and two orienting vector fields V_+ and V_-, which guide the robot to the target manifold.

The *nominal vector field*, V_n, is defined using the projection of V onto the constraint distribution, together with the angular rate of change of V induced by the direction of travel. Formally, for the constraint distribution $\mathcal{D}$ (defined

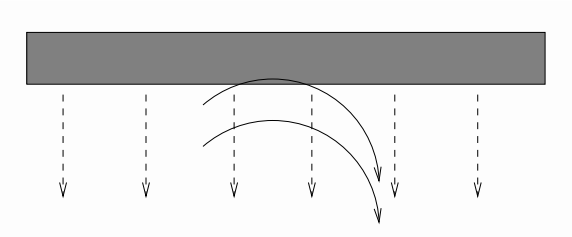

Figure 3.1: For any fixed gain, there exist initial states such that the robot crashes into the obstacle.

locally as the span of the tangent vectors to the manifold in which the system can move), the projection map $\mathcal{P} : \mathrm{T}Q \to \mathcal{D}$ is defined locally for some tangent vector $V(p) = v_x \frac{\partial}{\partial x} + v_y \frac{\partial}{\partial y} + v_\theta \frac{\partial}{\partial \theta}$ as

$$V_n(p) := \mathcal{P}(V(p)) = v_f \cos\theta \frac{\partial}{\partial x} + v_f \sin\theta \frac{\partial}{\partial y} + v_\theta \frac{\partial}{\partial \theta},$$

in which $v_f = \sqrt{v_x^2 + v_y^2}$ and $v_\theta = V_n(q) \cdot \nabla\theta_{V_n}(q)$. Note that on the target manifold, $V_n = V$. The orienting vector fields are defined in the obvious way:

$$V_+ = +\frac{\partial}{\partial \theta}, \qquad V_- = -\frac{\partial}{\partial \theta}.$$

It is now possible to define a global vector field that avoids the obstacles, satisfies the nonholonomic constraints, and whose integral curves are smooth and converge to the goal region:

$$W(p) = \begin{cases} -b(s)V_n(p) + (1 - b(s))V_+(p), \\ \qquad\qquad \theta_t(p) + \pi/2 < \theta \leq \theta_t(p) + \pi, \\ b(s)V_n(p) + (1 - b(s))V_-(p), \\ \qquad\qquad \theta_t(p) \leq \theta \leq \theta_t(p) + \pi/2, \\ b(s)V_n(p) + (1 - b(s))V_+(p), \\ \qquad\qquad \theta_t(p) - \pi/2 \leq \theta < \theta_t(p), \\ -b(s)V_n(p) + (1 - b(s))V_-(p), \\ \qquad\qquad \theta_t(p) - \pi < \theta < \theta_t(p) - \pi/2, \end{cases}$$

in which $b(s)$ is the bump function. We also need to define $s = \theta_{min}^{-1} \times \min\{|\theta - \theta_t|, |\theta - (\theta_t + \pi)|\}$. The quantity θ_{min} is defined below.

The vector field W blends between the nominal vector field and the orienting vector fields. It respects both the original target manifold and a copy of the target manifold shifted by π in the θ dimension. This is because the robot can follow the nominal vector field driving backwards from the orientation implied by the nominal vector field. Note that $W = V_n$ when the robot is on the target manifold, because $b(s) = 0$; also, we see that either $W = V_+$ or $W = V_-$ when $|\theta - \theta_t| = \theta_{min}$. This means that if the angular difference between the actual and target orientations is large enough, the vector field acts solely to orient the robot and does not translate it at all.

There are some requirements for the parameter θ_{min}. First, $\theta_{min} \leq \pi/2$. At this point, $V_n = 0$ because the original vector field is orthogonal to the orientation of the robot. Second, recall that the method presented in Chapter 2 guarantees that at the obstacle boundary, the computed vector field will satisfy the dot product constraint $V \cdot n_o > 0$, in which n_o is a unit normal vector pointing

away from the obstacle face. For V also having unit length, this implies that $\theta_{min} \leq \pi/2 - \cos^{-1}(\epsilon)$ in order for safety criteria to be satisfied. Since V can be constructed so that $\epsilon = 1$, this implies that if such a vector field is used as the target vector field, then it is possible to set $\theta_{min} = \pi/2$. Finally, it is obvious that θ_{min} must be greater than zero. Apart from these restrictions, the choice of θ_{min} is free.

A choice of small θ_{min} corresponds to an emphasis on actually following the original vector field; in the limit as $\theta_{min} \to 0$, we have the case where if the robot is initialized at an angle different from θ_t, it performs an orientation maneuver and then switches to following the target vector field. A choice of large θ_{min} corresponds to following the nominal vector field only as much as necessary to reach the goal eventually. The latter approach is interesting from a philosophical point of view because it is not at all important to follow the original vector field; the only thing that matters is that the robot reaches the goal. Rather than being a "desired" controller that we wish to track, the original vector field on $\mathcal{E}$ is simply an intermediate step which enables us to achieve the goal of building a smooth feedback plan over the whole configuration space.

Theoretical properties

It remains to verify that the vector field W possesses all the properties we desire.

Theorem 3.1 *The vector field W is smooth except for a set of measure zero and has smooth integral curves.*

Proof: The base vector field V is smooth except on the vertices of the convex cells and some edges that are never crossed by integral curves; so is its projection onto the constraint distribution, V_n. Similarly, the orienting vector fields V_+ and V_- are smooth. These vector fields are interpolated using a smooth bump function

b; the parameter s is smooth except where $|\theta - \theta_t| = \pm\pi/2$, a zero measure set. Similarly, the bump functions ensure that W is smooth except for $|\theta - \theta_t| = \pm\pi/2$. This simply corresponds to the initial choice of whether the robot will converge to the nominal vector field moving forward or backward (with an appropriate angular shift of π if moving in reverse). Therefore, the nonsmooth portions of W are a set of measure zero, and all integral curves flow away from the nonsmooth portions and are consequently smooth. ■

Theorem 3.2 *The vector field W satisfies the nonholonomic constraints of the robot at every point.*

Proof: The vector field W is constructed via the smooth interpolation of the three vector fields V_n, V_+, and V_-. Each of these vector fields satisfies the nonholonomic constraints: V_n since it is generated by the projection $\mathcal{P}$; V_+ and V_- directly by construction, as $\mathcal{P}(V_+) = V_+$ and $\mathcal{P}(V_-) = V_-$. Hence, the linear sum of these vector fields (through the interpolation) also satisfies the constraints. ■

Theorem 3.3 *The integral curves of W never lead to obstacle collision.*

Proof: The integral curves of the original vector field V never lead to obstacle collision; moreover, V is guaranteed to satisfy the obstacle constraint $V \cdot n_o \geq \epsilon > 0$ at the obstacle faces, as discussed above. Since we choose $\theta_{min} \leq \pi/2 - \cos^{-1}(\epsilon)$, we conclude that $b(t) = 0$ whenever the projected vector field V_n points into an obstacle face. This means that for any such point, $W = V_+$ or $W = V_-$, neither of which can cause collision since they induce rotation only. Consequently, if moving forward would cause a collision, the robot will rotate without translation until it is safe to translate once again. ■

The final result is that the integral curves of W converge to the goal region. This is a fairly obvious conclusion, given that the target manifold is attractive

and that the integral curves on the target manifold converge to the goal region. We arrive at the conclusion via two lemmas.

Lemma 3.1 *The integral curves of W converge to the target manifold. In other words, for any $\epsilon > 0$, $\exists T$ such that for all $t > T$, $|\theta(t) - \theta_t(t)| < \epsilon$.*

Proof: For any point $p = (x, y, \theta)$, consider the error in the θ coordinate, $d(p) = 1/2(\theta - \theta_t(p))^2$, in which $\theta_t(p)$ is the angle such that $(x, y, \theta_t(p)) \in M_t$. This has a unique minimum on the target manifold, where $\theta - \theta_t(p) = 0$. Moreover, since there are two target manifolds separated by π (one corresponding to moving forward and the other to moving backward), we may assume that $\theta \in [\theta_t(p) - \pi/2, \theta_t(p) + \pi/2]$. Therefore, $d(p) \leq \pi^2/8$ for all p.

From the construction of W, we can compute the time derivative of the distance function along any integral curve of the vector field: $\dot{d} = -(1 - b(s))|\theta - \theta_t|$, in which b is the bump function. The derivative of the distance function is negative everywhere except on the target manifold, because $b(s) = 1$ only on the target manifold, and $|\theta - \theta_t| > 0$ except when $\theta = \theta_t$, which is on the target manifold. Therefore, the angular error of the robot's trajectory to the target manifold asymptotically decreases to zero along any integral curve of the vector field W. Figure 3.2 illustrates this. ■

Lemma 3.2 *Consider two maximal integral curves $c_1(t)$ and $c_2(t)$ restricted to a cell other than the goal cell, with c_1 lying in the target manifold. Then there is at most one t_0 such that the coordinate projections $x(c_1(t_0)) = x(c_2(t_0)$ and $y(c_1(t_0)) = y(c_2(t_0))$.*

Proof: Essentially, this lemma means that if the curves in a cell are projected into the (x, y)-plane, they can cross at most once. In the plane, the curve c_1 partitions the cell into two halves; denote these "left" and "right" (see Figure

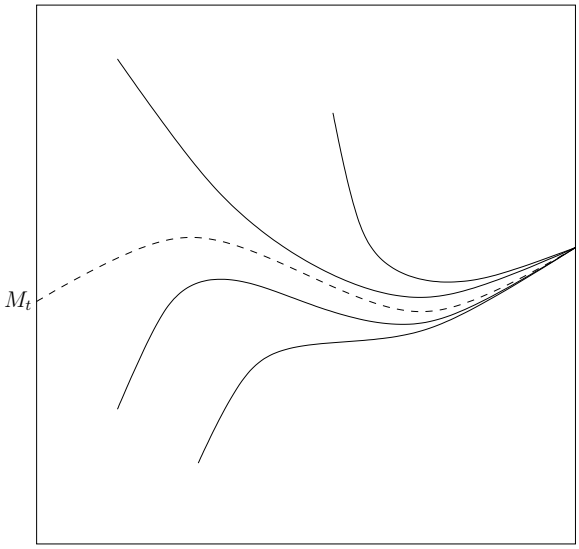

Figure 3.2: An illustration of convergence to the target manifold, M_t.

3.3 for an illustration). Crossing c_1 from left to right corresponds to $\theta < \theta_t$, and crossing from right to left to $\theta > \theta_t$. This means that if the curves cross twice, they must cross at one time with $\theta > \theta_t$ and one time with $\theta < \theta_t$. This can only happen if at some point the curve c_2 crosses the target manifold. But by construction, no curve crosses the target manifold, since the vector field is identically equal to the nominal vector field on the target manifold. In fact, no curve ever actually reaches the target manifold, although all integral curves approach it tangentially. Hence, c_2 can cross c_1 at most once. ■

Theorem 3.4 *The integral curves of W converge to the goal region.*

Proof: First, consider the case of an integral curve over some cell in $\mathcal{E}$ other than the goal cell. Using argumentation similar to Chapter 2, we will see that the integral curve will exit that cell via the exit face of the cell. Recall that the cell is partitioned into different regions using the GVD. The construction of V guarantees that if an integral curve of V crosses a face separating two regions of the cell, then it will not cross that face again. Since we know that a flow from

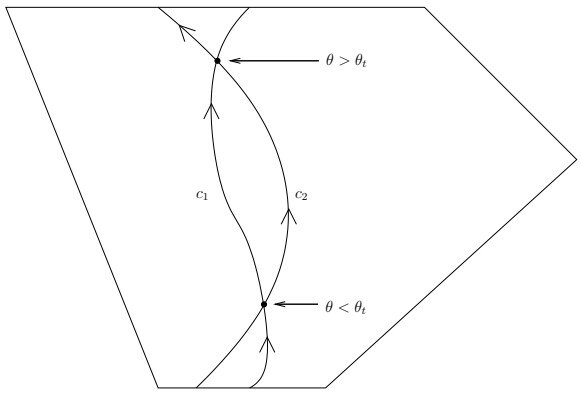

Figure 3.3: An impossible situation. The integral curve c_1 partitions the cell into two halves; crossing from one side to the other at a given point implies a relationship between the relative orientations at that point.

W cannot cross any flow from V more than once (Lemma 3.2), this implies that the flow from W cannot cross a separating face more than once. Therefore, there are no cycles that cross the GVD; integral curves must either exit the cell or get trapped in a region corresponding to a particular face. However, we know that the flows from V do not get trapped because they satisfy a dot product constraint $V \cdot n_s > 0$ with the normal vector of the hyperplane separating the face from the exit face. The convergence of the angle θ as shown in Lemma 3.1 implies that for any ϵ, there exists a time T such that $W(t) \cdot V(t) \geq 1 - \epsilon$ for all $t \geq T$. Together with the corresponding fact that $||W(t)||$ converges to 1, this implies that there is also some time T_1 such that $W(t) \cdot n_s > 0$ for all $t \geq T_1$. Hence, by the same reasoning as in Chapter 2, the integral curve cannot be trapped and must exit the cell by the exit face.

Just as all integral curves of W must reach the goal cell, all integral curves in the goal cell converge to the goal region $x_g \times S^1$. The original vector field V satisfies the condition $V \cdot (x_g - p) > 0$ at every point p. Exactly as argued above, we know that there exists a time T such that $W(t) \cdot V(t) \geq 1 - \epsilon$ for all $t \geq T$. This

leads to the conclusion that there exists a time T_1 such that $W(t)\frac{d}{dt}(x_g - p) > 0$ for all $t > T_1$.

Since for all nongoal cells the integral curves all properly exit via the exit face (after sufficient time), and all integral curves in the goal cell converge to the goal region, the integral curves globally converge to the goal region. ■

3.3.2 Analysis and examples

From a practical standpoint, this method is highly advantageous. In addition to the desirable attributes of the system trajectories our feedback strategy induces, the vector field is extremely fast to compute at any point. The extension to unicycles has the same algorithmic complexity and practical time requirements as discussed above. First, the component vector fields must be computed for the given polygon. If the environment is given as a general PL environment, then it must first be decomposed into convex pieces. Since it is just two dimensions, this is algorithmically straightforward and extremely fast.

This implies that the vector field is extremely fast to compute, even for very large environments. A large environment requires more preprocessing than a small one, but the execution in real time is no different. Our method is entirely suitable for real time feedback control. If the environment is dynamic, not static, our method is still efficient in practice. If the change to the environment is sufficiently local that the environment polygon and connectivity graph do not have to be recomputed, then our method incurs no extra cost. In the worst case, the convex decomposition must be reperformed; however, even this operation is fast enough that it can be done with no noticable drop in performance for reasonable environments.

For the sake of illustration, several examples of the system trajectories of this algorithm are included. There is a substantial amount of design freedom in choosing how aggressively to approach the target manifold. Figure 3.4 shows several trajectories produced by our algorithm. Starting from an initial point in the plane, the initial orientation of the robot is set to ± 0.5 radians from the angle of the original vector field at that point. The trajectories of the aligned system are shown by dashed lines, and trajectories corresponding to more or less aggressive tracking strategies are shown in the solid lines.

Another example is given in Figure 3.5, which indicates the orientation of the robot at different points along its trajectory.

3.4 Point Robots with Bounded Curvature Constraints

In this section, we will consider a more difficult system than the unicycle described above: the smooth feedback planning problem for car-like robots. The input, once again, is a PL environment and a goal state, x_g. We will describe an algorithm to construct a feedback controller (equivalently, a vector field) such that from any initial state, following the integral curves of the vector field causes the robot to asymptotically converge to the goal state. Moreover, the algorithm computes a *smooth* feedback controller, which guarantees that all system trajectories are C^∞ differentiable.

Car-like robots are characterized by a bounded curvature constraint. This can be expressed as a constraint on the angular velocity: $|\dot{\theta}| \leq \alpha v_f$, in which $\alpha > 0$

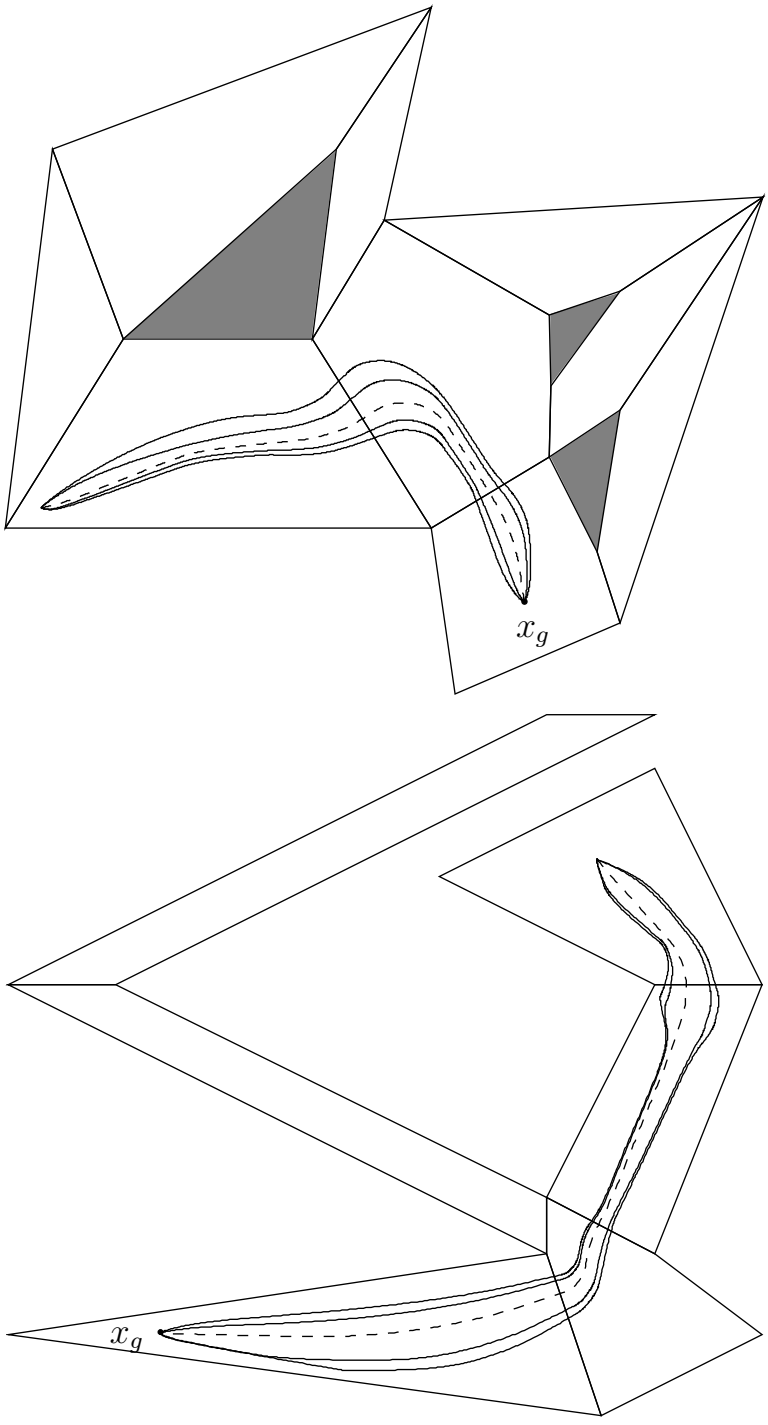

Figure 3.4: Two environments, with goal states x_g and trajectories from an initial point with initial angular deviation.

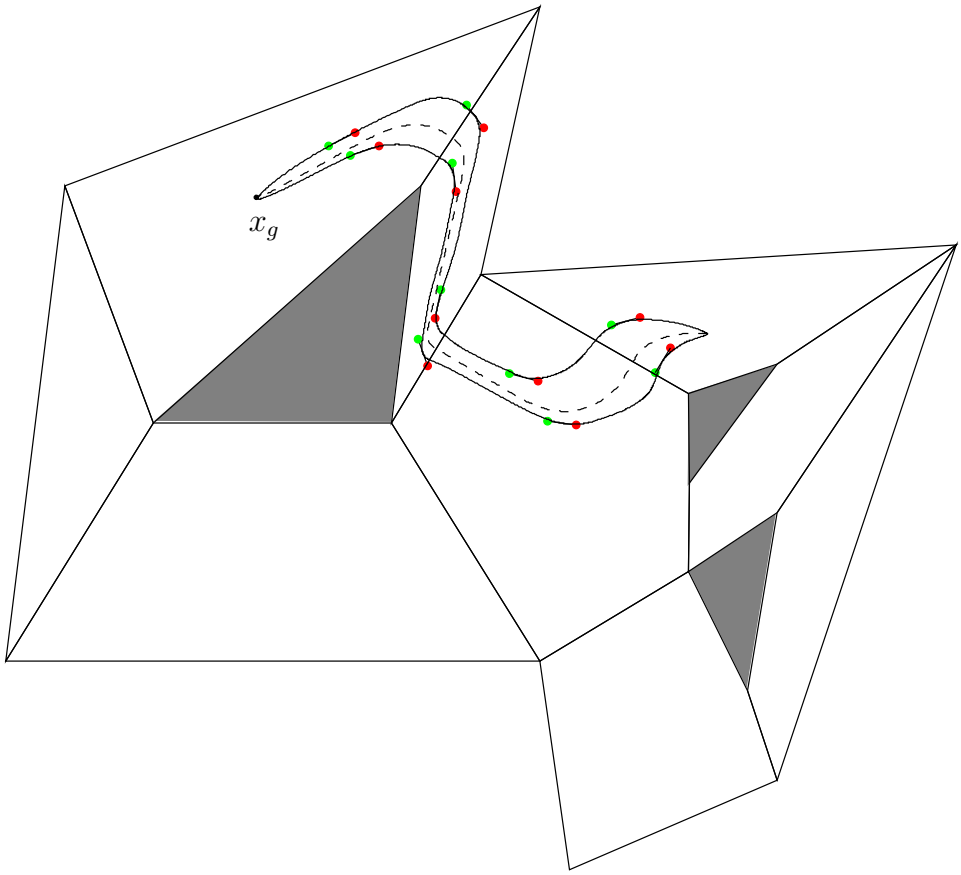

Figure 3.5: Two trajectories from an initial point, with indications of robot orientation.

and v_f is the forward velocity. The state transition equation for the system is then

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{v}_f \end{pmatrix} = \begin{pmatrix} v_f \cos\theta \\ v_f \sin\theta \\ v_\theta \\ u_f \end{pmatrix}, \tag{3.4}$$

in which we again have $|v_\theta| \leq \alpha v_f$. This is illustrated in Figure 3.6. The control variables for this model are v_θ and u_f. Car-like robots pose a much more challenging control problem than differential drive (unicycle) robots due to the bounded curvature constraint. A unicycle is capable of going around a sharp corner without any difficulty because of its ability to rotate in place. A car-like robot, however, may need to reverse directions a number of times to move through an area with small clearance. The classic example of this is the problem of parallel parking a vehicle in a tight space.

Note that for car-like robots, there is no smooth (or even continuous) vector field over the configuration variables (x, y, θ) alone that can reverse the velocity of the vehicle. Any such trajectory could at best converge to $v_f = \dot{\theta} = 0$, since $v_f = \dot{\theta} = 0$ is an equilibrium, and there is no way to go from $v_f > 0$ to $v_f < 0$ except through the equilibrium. Taking v_f as a state variable allows us to define smooth controls that reverse velocity. In the examples in Section 3.4.2, what may appear to be "cusps" in the configuration space are actually smooth curves in the expanded (x, y, θ, v_f) space. Although v_f is included in the state space, note that there are no bounds on the inputs. This is important for obstacle avoidance because there is no way to guarantee that the obstacles will be avoided without having unbounded inputs as the robot approaches the obstacle boundary.

Another notable feature of this model is the absence of a steering angle. In standard car models, the angular velocity is typically $\dot{\theta} = v_f / L \tan\phi$, in which L

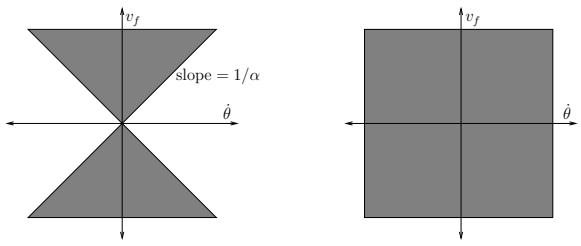

Figure 3.6: Angular versus linear velocity, for a car-like robot (left) and a unicycle (right).

is the length of the car (i.e., the distance between the front and rear wheels), and ϕ is the steering angle. In our model, however, the robot is a point, and $L = 0$. As a result, steering angle ceases to be meaningful. That said, an implicit steering angle can be computed from the v_θ determined by our feedback controller, given some $L > 0$.

Brockett's condition [199] implies that no static, smooth vector field (feedback control) can stabilize the complete state of the system; therefore, only partial state stabilization is attempted. If desired, other controllers can be used to stabilize the full state of the system after the robot reaches a sufficiently small neighborhood of the goal state. This is still a significant result, because other approaches are either open loop (losing the robustness of feedback control) or cannot guarantee obstacle avoidance as well as global asymptotic convergence.

3.4.1 Feedback plans for car-like robots

In this section, smooth feedback plans for car-like robots amidst obstacles are described. As in Chapter 2, the strategy is as follows:

1. Decompose the environment into convex cells and compute a discrete plan over the cell connectivity graph.

2. Define local controllers (vector fields) corresponding to each cell and each face separating adjacent cells.

3. Construct a global controller by interpolating between vector fields defined in each cell.

By using a smooth interpolating function, smoothness can be guaranteed, not only in the interior of each cell, but also across cell boundaries.

Constructing a smooth feedback plan

First, the environment (a polygonal subset of $\mathbb{R}^2$) must be decomposed into convex cells and a discrete plan computed. Once the decomposition has been computed, consider the connectivity graph of the cells which is the dual of the decomposition. Let the cell containing the goal point, x_g, be denoted as C_g. Then, beginning with C_g, search the connectivity graph to obtain a chain of cells from any cell to C_g.

The remaining task is to construct local controllers that avoid obstacles, are consistent with the computed discrete plan, and satisfy the smoothness requirement. Since each node in the graph corresponds to a convex cell, consistency with the high level plan is equivalent to solving the *control to facet* problem: that is, all integral curves must exit from a particular facet in finite time while avoiding all other facets. These controllers are constructed by defining a vector field over the cell, as well as one corresponding to each face. In the interior of the cell, we will interpolate between these vector fields in such a way that all the requirements are met. In particular, the vector field at a point p in a particular cell is defined as

$$V(p) = b(p)V_i(p) + (1 - b(p))V_c(p), \tag{3.5}$$

in which b is the interpolating function, V_i is the vector field corresponding to

some face f_i, and V_c is the vector field corresponding to the cell. Interpolation is once again accomplished using the generalized Voronoi diagram (GVD).

Now, we will describe the cell and face vector fields in detail. The vector field corresponding to the cell, V_c, contains only a rotational component. The sole purpose of the rotational vector field is to orient the robot so it may freely cross the exit face into the next cell, without encountering another face first. Assume that we are in an intermediate cell (the case of the goal cell will be considered later), and that we are given a point on the exit face of the cell; such a point is trivial to compute. Define θ_e as the angular error of the robot to the point, given the current orientation. The absolute angle to the point from the robot is $\theta_p = \text{atan2}(y_p - y, x_p - x)$. We then have $\theta_e = \min\{\theta - \theta_p, 2\pi - \theta + \theta_p\}$. Clearly, this has a maximum of π when the robot is pointing directly away from the point and a minimum of zero when the robot is oriented directly at the point. Let $\theta_e = \theta - \theta_p$. For any simple control law of the form $\dot{\theta} = -K\theta_e = -K(\theta - \theta_p)$ with $K > 0$, it is clear that θ_e will remain less than π for all time and therefore $\theta_e = \theta - \theta_p$ always (a transition to $\theta_e = 2\pi - \theta + \theta_p$ is never made). The importance of this fact will be made clear. Define the cell vector field:

$$V_c = -\alpha |v_f| \text{sgn}(\theta_e) b(\theta_e/\epsilon) \frac{\partial}{\partial \theta} \tag{3.6}$$

for some $\epsilon > 0$, with b the bump function above and v_f the velocity. The presence of αv_f in the product guarantees that the bounded curvature constraint is satisfied by this vector field. Also, θ_e never switches between terms in the min expression, which is important for preserving smoothness. Finally, the sign of θ_e never changes, as we argued above; this is also necessary for smoothness. The effect of the rotational vector field is to orient the robot toward the point p on the

exit face. Even though the value of θ_e is not monotonically decreasing, it should be clear that it will eventually converge to zero; this will be proven later.

Now we consider the vector fields $\{V_i\}_1^n$ corresponding to the faces $\{f_i\}_1^n$. The face vector fields will have no rotational component, but will consider only the forward velocity. The purpose of the face vector fields is to prevent the robot from reaching any face other than the exit face (by reversing the velocity of the robot). We can construct each face vector field using two other vector fields; one to decelerate the robot and prevent it from hitting the face, and one to accelerate the robot in the opposite direction.

A couple of preliminary definitions are required. Assume that the point $q = (x, y, \theta, v_f)$ is in the region of influence of f_i (i.e., the Voronoi region corresponding to f_i). Then, define the *hitting time* t_h to be the time until the robot hits the f_i, while maintaining the current heading and forward velocity. If the integral curve through q does not hit f_i but leaves the region of influence of f_i, let $t_h = \infty$. Note that the integral curve containing q does not depend on the velocity v_f, since the rotational component defined by V_c is proportional to v_f; thus, whether or not the integral curve hits f_i can be determined without considering V_i. Define the saving vector field:

$$V_s(q) = -(v_f/t_h)\frac{\partial}{\partial v_f}. \tag{3.7}$$

Observe that if the robot travels in a straight line, this is sufficient to stop the robot before the edge is reached, assuming $|v_f| \leq 1$.

In addition to the saving vector field, we need a vector field over the entire cell that accelerates the robot away from the face. This will increase the forward velocity if the robot is moving away from the face (i.e., $t_h = \infty$) and slow the robot down if it is moving toward the face ($t_h < \infty$). Let $V_+ = \text{sgn}(v_f)(1 - b((|v_f|/\epsilon) - (1 - \epsilon)))$; this vector accelerates the robot to its maximum speed of one. Also

define $V_- = -\text{sgn}(v_f)$, which decelerates the robot. Let θ_i be the absolute value of the angle between the robot's velocity and the inward pointing normal of face f_i; then define the acceleration vector field:

$$V_a(q) = b((\theta_i/\epsilon)V_+(q) - (1 - b((\theta_i + \epsilon)/\epsilon))V_-(q). \tag{3.8}$$

If the integral curve does not hit the face (with $t_h = \infty$ as a result), simply let $V_a(q) = V_+(q)$. Simply, this vector field decelerates if $\theta_i < 0$ and accelerates if $\theta_i > 0$, smoothly interpolating between the two in the interval $(-\epsilon, \epsilon)$.

The face vector field for face f_i is then defined as

$$V_i = b(t_h - t_{safe})V_s(q) + (1 - b(t_h - t_{safe}))V_a(q) \tag{3.9}$$

for some $t_{safe} > 0$. The face vector field smoothly interpolates between the saving vector field (when collision is imminent) and the ordinary acceleration vector field (when collision is at least t_{safe} time from occurring). The only exception is when the face is the exit face, in which case we set $V_i = V_a$, since we want the integral curve to reach the exit face.

We have seen how to construct vector fields over intermediate cells; now, consider the case of the goal cell. In this case, the target point is the goal point x_g which is in the interior of the cell, rather than on the boundary as in the previous case. This would seem to complicate things because it is no longer trivial to argue that rotating in the same direction will lead to orientation toward the target point. Since x_g is in the interior of the cell, it is possible for the integral curves to converge to a circular orbit about the goal point. The most straightforward way to remedy this is to partition the goal cell into new cells such that the goal point is once again on the boundary. This is easily accomplished; partition the

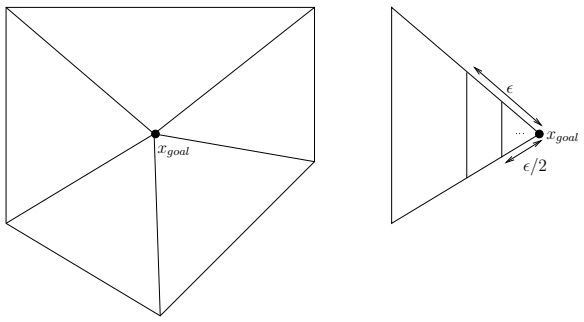

Figure 3.7: A goal cell, subdivided. On the right, an infinite sequence of cells is created, each of which is guarantee that the robot is within $\epsilon/2^i$ for some i.

cell into triangles, one per face, the vertices of which are the vertices of that face together with the goal point. Then, for some $\epsilon > 0$, add a new edge with vertices on the two edges incident on x_g and ϵ distant from it. Thus, crossing that edge ensures that the robot is no more than ϵ away from the goal point. At this point, the vector field is constructed exactly as has already described. This procedure is illustrated in Figure 3.7. Inside the new "corner" cell, the procedure can be repeated by trimming off the corner at distance $\epsilon/2$, and so on. This will guarantee global convergence to the goal state (in position, not orientation).

Theoretical analysis

To establish that the algorithm correctly computes feedback plans, several important results must be proven. First, we will verify that the controller satisfies the bounded curvature constraints. Second, it is straightforward to show that the controller is safe, avoiding all obstacles. Third, we will show that all integral curves of the global vector field V are smooth. Finally, we will prove that the controller stabilizes the position of the robot. These results directly follow from the construction of the vector fields over the individual cells.

Theorem 3.5 *The vector field V satisfies the bounded curvature constraint.*

Proof: Our two control inputs, as determined by V, are $\dot{v}_f$ and $\dot{\theta}$. From the definition of the cell vector field V_a, we see that $\dot{\theta} \leq \alpha v_f$, which is precisely the bounded curvature constraint. ■

Theorem 3.6 *The controller defined by V avoids all obstacles.*

Proof: Each obstacle face is a face on one of the cells in the convex decomposition. Consider first the case of a trajectory approaching a point in the interior of a face f_i. From the construction of V, we see that $V = V_i$ on the face. Therefore, we need only consider the face vector field V_i. As the robot approaches the face, the hitting time t_h goes to zero and $V_i \to V_s$, the saving vector field. We have already mentioned (and it is trivial to verify) that the saving vector field is always strong enough to reverse the velocity before the face is reached. Another important case is when the integral curve approaches an endpoint of the face, rather than a point in its interior. It is somewhat more difficult to show that safety is preserved in this case; the results will not be given here, but it is always possible to choose the saving vector field V_s large enough that obstacle collision is avoided in this case as well. ■

Theorem 3.7 *All integral curves of V are smooth.*

Proof: The bump function b is smooth, as is the analytic switch we use to interpolate between the face and cell vector fields in each face's region of influence. So we simply need to consider the smoothness of the cell and face vector fields. Consider the face vector fields. If the velocity-independent integral curve does not lead to the face, then we have $t_h = \infty$ and the velocity along that integral curve smoothly increases to the maximum velocity, $|v_f| = 1$. Consider, then, the integral curves along which the system must reverse direction so that the face is not reached. In this case, the vector field is a smooth interpolation of the saving vector field V_s and the acceleration field V_a. The saving vector field is clearly

smooth, since the hitting time and velocity are smooth. The only potential issue is when the the hitting time goes to infinity, which happens when $v_f = 0$ or θ becomes parallel to the face. At this point, however, we have $V_i = V_a$ and all derivatives of the bump function are identically zero; therefore, smoothness is preserved. Likewise, V_a is also smooth.

Now, consider the cell vector field, which controls rotation. The velocity v_f is smooth, as we have shown above. We have also discussed the fact that θ_e is smooth, since the system always rotates in the same direction. Therefore, the cell vector field V_a is smooth. Therefore, all component functions and vector fields are appropriately smooth, so all integral curves of V are smooth. ■

Theorem 3.8 *All integral curves of V converge to the goal state, x_g.*

Proof: To show this, we will prove that for any cell, all integral curves will reach the exit face of that cell in finite time. After a number of such faces are crossed, we can guarantee that the robot is at most $\epsilon, \epsilon/2, \epsilon/4, \ldots$, distant from the goal state, which establishes convergence.

First, we can verify that when the robot changes direction to avoid hitting a face, it actually changes direction rather than simply converging to a point. The robot will not converge to a point under the acceleration vector field V_a, because the only place where $V_a = 0$ is when $\theta = \theta_i$. The only time when $v_f = 0$ is when the robot reverses direction, but it is not possible for θ to equal θ_i at this point, because then the robot would not be at risk of hitting the face (i.e., it would proceed to another face's region of influence instead). Therefore, the robot would not decelerate and change direction. The case where $v_f = 0$ and $\theta = \theta_i$ does correspond to an equilibrium point, but one which has no region of attraction (if the system starts at this point, perturb it; otherwise, no integral curve converges to the state).

Consider, then, the saving vector field V_s. The term v_f/t_h implies that the deceleration applied is independent of the magnitude of the velocity, since t_h depends linearly on v_f. Since the deceleration is velocity-independent, this will never lead to the robot converging to a point.

One potential problem is on the faces of the GVD, where $V = V_c$. There is no linear acceleration on the GVD face since V_c is purely rotational, and since V_c depends linearly on v_f, any point on the GVD such that $v_f = 0$ is an equilibrium point. However, it can be seen that these have no region of attraction, since any perturbation will lead away from the equilibrium region. This means that the robot will never converge to this region. If the robot is in this location due to an initial condition, a random perturbation will free it from the equilibrium, so there is no problem. Note that while we do not describe it here, it is possible to add a linear acceleration component to V_c which will eliminate this issue entirely.

Finally, it must be shown that the robot will eventually be oriented toward the target point enough that it will be able to leave via the exit face of the cell. Recall that we have already discussed the fact that within a cell, the robot will always rotate in the same direction, due to the cell vector field. Define a *trajectory segment* to be an interval of an integral curve between two velocity reverals. Consider a sequence of trajectory segments t_1, t_2, t_3, t_4. Denote by $e_1, \ldots, e_4$ the linear extrapolation of the endpoints on the faces of the polygon. Assuming that we are rotating in the positive direction, we know that e_3 is located counter-clockwise of e_1 along the boundary of the polygon, and e_4 is likewise counter-clockwise of e_2. Taking then a sequence of every other endpoint, we can see that this must eventually reach the exit face. There can be no limit point at any other point along the boundary, because this would imply that the angular change over the trajectory segments goes to zero, which is impossible. Therefore, the robot will

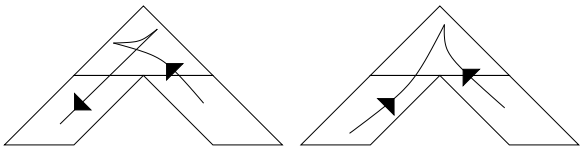

Figure 3.8: On the left, the robot makes an extra reversal but always crosses edges moving forwards. On the right, the robot crosses moving backwards but saves a reversal.

always rotate in such a way to be oriented toward the exit face and therefore exit the cell via that face. ■

3.4.2 Extensions and practical results

This feedback planning algorithm can be improved and extended in a number of ways. In Section 3.4.1, we defined the cell vector field in such a way that the robot always rotates in the same direction. This was utilized in the convergence proof above. Practically, this means that the robot will leave a cell with a velocity identical in sign to the velocity it entered with. This means that if the robot enters the cell moving forward, it will exit the cell moving forward. This is desirable behavior, if there is a practical difference between forwards and backwards for the robot. If there is not such a distinction, then it may be possible to improve path quality by allowing the robot to reverse the direction with which it crosses the edge. This can be seen in Figure 3.8. In order to do this, the robot may no longer always rotate in the same direction, but may reverse the direction of rotation when the robot is moving away from the target point. Following this strategy may lead to many fewer path reversals, especially when the robot must go around sharp corners.

A second way to modify the vector field is to permit the robot to reverse direction even when a collision with the edge is not imminent. For example, if the robot is moving away from the exit face but oriented toward it, it is advantageous

for the robot to immediately reverse direction rather than waiting until an edge is approached. We have not investigated this approach at present.

Although our method defines a global feedback plan, it can also be used to generate open loop trajectories. In fact, dynamic programming can be used to reduce considerably the number of path reversals in an individual open loop trajectory. Consider an initial state x_i. If there are n possible exit edges, then there are n possible choices of cell vector field; add an open node corresponding to each choice to a priority queue. For each of these, the integral curve through the initial state will reach some face, either crossing it or reversing velocity. For each possibility, insert a new node in a priority queue with cost equal to the number of reversals so far (in the first iteration, the node corresponding to crossing the edge would have lower cost with zero reversals, and the state remaining in the current cell would have high cost with one reversal). Each time a cell is reached, new nodes are added based on the number of possible exit edges. Using dynamic programming, continue to extend each trajectory segment, adding new elements to the queue every time a cell boundary is reached. Once the goal is reached, we can guarantee that the minimum number of reversals has been achieved for that open loop query, from the class of all possible controllers constructed according to our method.

Finally, several examples of paths computed using this method are presented in Figures 3.9 and 3.10. These figures depict only two-dimensional projections of individual integral curves; however, the method determines a global feedback plan over the entire four-dimensional state space.

In some cases, it may seem that there are unnecessarily many path reversals. To some extent, reversals are unavoidable because of the bounded curvature restriction; several properties of our algorithm increase the number of path reversals. In particular, many reversals can be induced by the requirement that once a robot

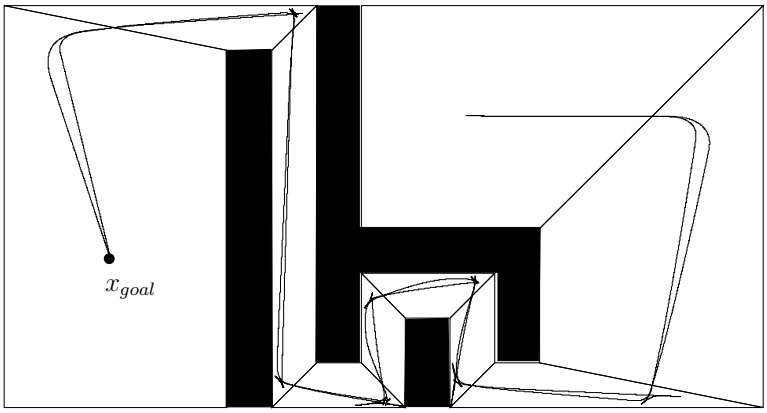

Figure 3.9: Paths through a winding corridor, for robots with different turning radii.

enters a particular cell, it cannot leave that cell except via the exit face (small cells greatly magnify this problem). Other factors that contribute to reversals are the choice of the target point and the fact that edges are crossed with zero angular velocity. It is simple to modify the choice of target point; for example, it could be replaced with a target interval on the exit face, which would still guarantee convergence but would not "compress" the integral curves to the target point as is seen in the examples (in the examples, the target point is the midpoint of the edge). It may be possible to add a rotational component to the face vector fields, which would likely improve path quality; however, global convergence under such a scheme has not been shown.

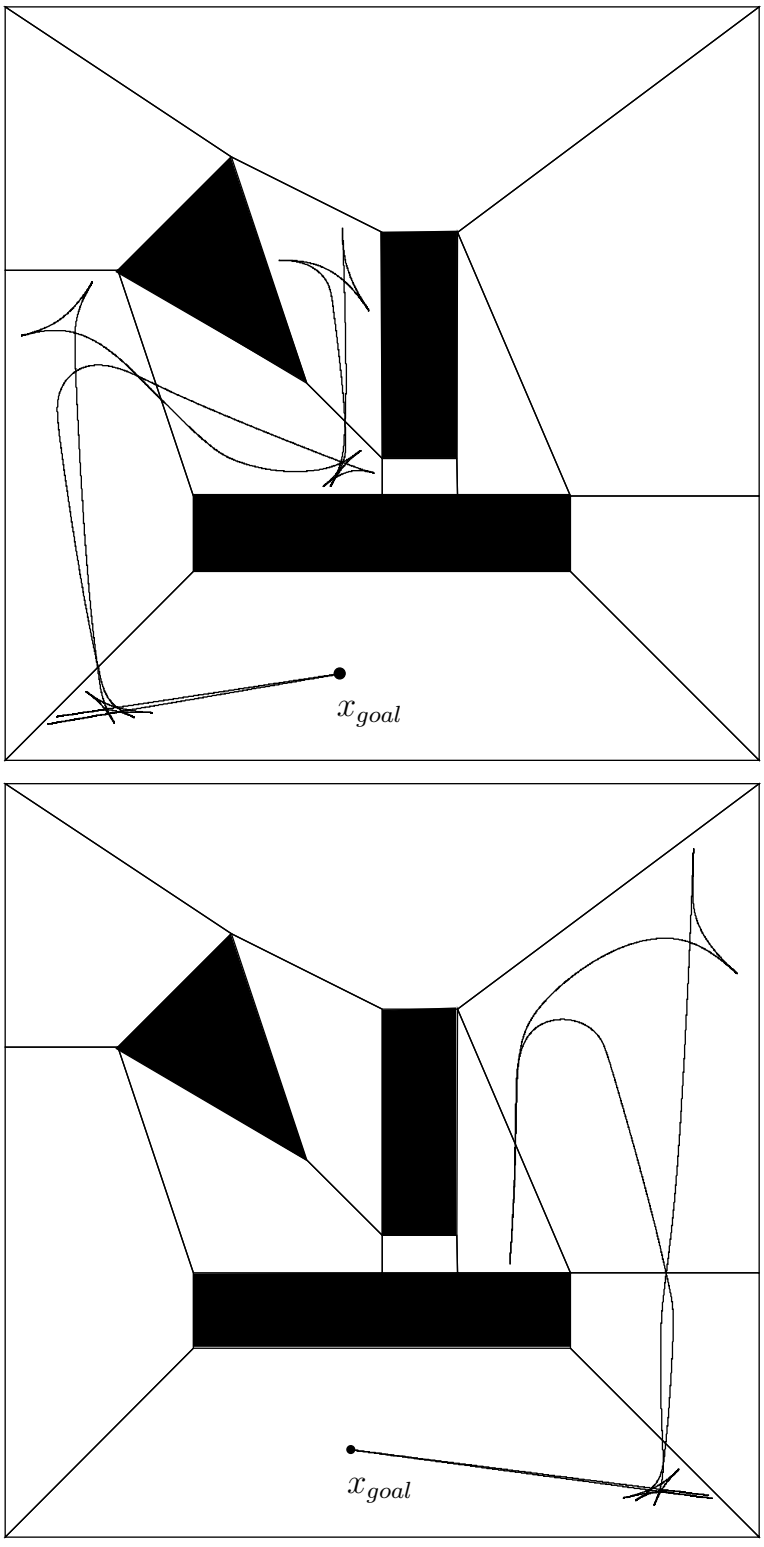

Figure 3.10: Paths through an obstacle cluttered environment, for two robots with different turning radii.

CHAPTER 4

FURTHER EXTENSIONS

4.1 Introduction

In the previous chapter, we described smooth feedback planning algorithms for two types of simple nonholonomic systems: unicycles and car-like vehicles. In addition to their practical relevance, these results are significant because they demonstrate that the vector field interpolation paradigm is applicable for systems other than the fully actuated ones described in Chapter 2.

The purpose of this chapter is to explore several additional results which can be easily attained using vector field interpolation. The main goal, once again, is to demonstrate the flexibility and usefulness of this approach for practical mobile robot tasks. First, we describe how to construct simple trajectory tracking controllers using vector field interpolation. Second, we briefly describe a centralized algorithm using vector field interpolation to solve the problem of multiple robot coordination. The results in this chapter will not be described in as much detail as in the previous chapters; rather, the intent is to present key ideas which may motivate future theoretical and practical work.

4.2 Trajectory Tracking

In this section, we will see how to construct smooth feedback plans which cause the robot to converge to a specified target *trajectory*, rather than to a goal point.

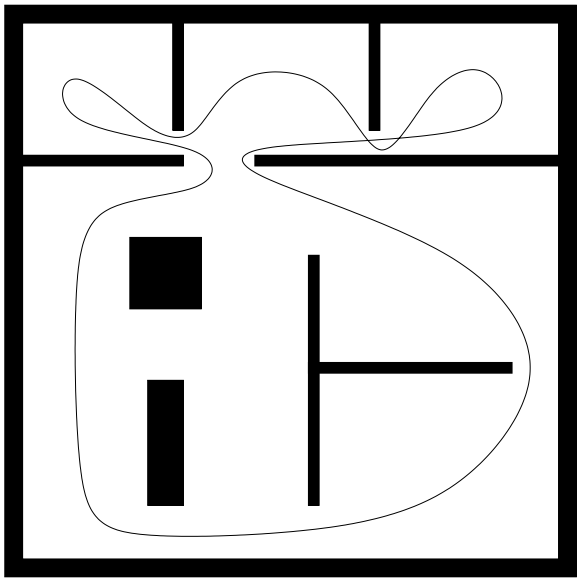

Figure 4.1: A robot patrol task might be specified as a curve through an environment.

This is an important problem for many mobile robot applications, because trajectories can often be used to encode the execution of robot tasks. For example, the trajectory might represent a safe path through a dangerous environment, or a route to patrol (see Figure 4.1). The trajectory could be provided by a human user; automatically generating trajectory tracking plans is one way to enable effective human-robot teams. Trajectory tracking controllers could also be useful for machining operations or manufacturing tasks. Using the vector field interpolation framework described in this dissertation, we can construct safe, obstacle-avoiding trajectory tracking controllers in a straightforward and natural way.

4.2.1 Background

Consider a robot task that involves following a specified trajectory $c(s), 0 \leq s \leq S$ through the workspace. This trajectory could be a closed loop (i.e., $c(0) = c(S)$), or it could have distinct endpoints. In classical control theory, a standard approach

is to parameterize the target trajectory by time [79, 80]. Using t as the parameter of the curve, we can write the error at time t as $e(t) = c(t) - x(t)$; the controller, then, should force $|e(t)| \to 0$ as $t \to \infty$.

Many have noted that this time dependence is often an unnecessary encumbrance for many robotics applications [130, 137]. It is generally more important for the robot to follow the path geometrically that to adhere to a strict time parameterization. If time dependence is unnecessary, then a trajectory tracking controller should simply cause $d(x(t), c) \to 0$ as $t \to \infty$, in which $d(x(t), c) = \min_s d(x(t), c(s)$ is the distance from $x(t)$ to the closest point on the curve c. The most trivial way to accomplish this, which is to choose any point on the curve (perhaps the current nearest neighbor) and stabilize the system to that point, is inadequate because it neglects the key requirement of the trajectory tracking problem: that the robot proceed along the path at some rate (albeit not not at a prespecified one).

A number of different approaches to solving this problem have been adopted in the literature, which can be divided into two types. One basic approach is to explicitly choose a reference point on the curve for any state of the robot; this might be defined as $r : X \times \mathbb{R} \to X$, taking arguments of state and time and returning the reference point. Alternatively, the reference point could be a simple robot configuration rather than a state. If the reference point function is chosen carefully, then the robot should converge to the path and proceed along it. The *dynamic path following* approach chooses the nearest point on the path as the reference point for any robot state and adds a component that induces a forward velocity along the path [200]. Egerstedt et al. choose a reference point using a "virtual vehicle" approach, in which the reference point moves along the curve according to a differential equation that involves error feedback [201].

A second approach is to define a vector field whose integral curves converge to the target trajectory. A prominent example is *velocity field control*, which has been used for control of manipulator arms and mobile robots [129–131,133,134,137,202]. In these approaches, a vector field (velocity field) over the workspace is defined such that following the vector field is equivalent to solving the task (following the trajectory). A passive controller for the system is then designed so that convergence to the velocity field is guaranteed.

This approach combines aspects of both of these approaches. On the one hand, the overall philosophy of this approach is certainly oriented toward vector fields. However, whereas velocity field control is less concerned with the construction of a task-solving vector field and more concerned with the passive controller, we are primarily concerned with constructing a vector field which provably solves the task and avoids obstacles. Similar to the previous work utilizing reference points, the convergence proof of this approach relies upon guaranteeing that the distance to a chosen reference point always decreases along integral curves of the constructed vector field.

4.2.2 Preliminary results

There are two key steps to constructing a vector field for tracking a specified trajectory in the plane. First, the decomposition needs to consider the curve as well as the environment; the curve segment within each cell should be simple enough that it is straightforward to define a vector field that causes the robot to get closer to the curve as it traverses the cell.

Once a good cell decomposition has been constructed, the vector field will be constructed by smoothly blending component vector fields. As one might expect,

these vector fields correspond to the intuitive requirements of (i) converging to the curve, and (ii) progressing along it.

Constructing a good decomposition

For a general smooth planar curve, the task of constructing an appropriate cell decomposition can be quite difficult. The problem is closely related to that of approximating the curve itself, so consider the problem of approximating a planar curve $c(s)$ with straight line segments. How should we determine the endpoints of the line segments so that the curve is approximated "well enough"? At a high level, this is clearly determined by the curvature of c. If the curve (or a region of it) has high curvature, then the curve must be approximated by many line segments; if it has low curvature, then a few line segments should suffice. Building a decomposition containing the curve is quite similar, with the exception that our goal is to construct a polygonal tunnel around the curve, rather than simply a piecewise linear approximation of it.

We may begin by introducing a few necessary definitions and assumptions. Let the parameter s correspond to the unit length parameterization of c. The (absolute) curvature $k(s)$ is defined as

$$k(s) = \left| \frac{dT(s)}{ds} \right|,$$

in which $T(s)$ is the tangent vector to $k(s)$. Recall that for a curve $c(s)$ with maximum curvature k_{max}, the radius of the smallest osculating circle of the curve is $1/k_{max}$.

Assume the following with respect to c:

1. The maximum absolute curvature of $c(s)$ is k_{max}, with $0 \leq k_{max} < \infty$.

2. There exists a number $d_{min} > 0$ such that:

   (a) $d_{min} < 1/k_{max}$,

   (b) The curve is separated from the obstacles by at least d_{min}, and

   (c) The curve is separated from itself by at least d_{min}.

The condition that the curve be separated from itself by d_{min} means that for any point $c(s_0)$, $|c(s_1) - c(s_0)| \leq d_{min}$ implies that $|c(s) - c(s_0)| \leq d_{min}$ for all $s \in [s_0, s_1]$. In other words, the curve never loops around in such a way that it is closer than d_{min} from a previous point on the curve. (Later, we will briefly address the case where the curve intersects itself, such as when the desired trajectory is a figure eight.)

An important implication of these assumptions is that for any point s_0 on the curve and ball of radius d_{min} centered on $c(s_0)$, $B(c(s_0), d_{min})$, the line normal to the curve at s_0 partitions the curve such that one half of $B(c(s_0), d_{min})$ contains only curve points $c(s)$ for $s > s_0$, and the other half contains only curve points $c(s)$ for $s < s_0$. This is illustrated in Figure 4.2. The curve enters the ball on one side of the line, passes through $c(s_0)$, and exits the ball on the other side of the line. The curve does not cross the line inside the ball except at $c(s_0)$, and never enters the ball apart from the interval (s_a, s_b). The separation requirement guarantees that the curve never reenters the ball; that the curve does not cross the normal line except at s_0 is guaranteed by the condition $d_{min} < 1/k_{max}$, since that would imply a violation of the maximum curvature assumption.

Now, we may consider the construction of a polygonal tunnel that contains c, lies entirely in the free configuration space, and is sufficiently simple that defining a convergent vector field will be easy. First, partition c into regions with sign-invariant curvatures, and further partition the curve so that for each region, the maximum difference in $\phi(T(s))$ (the angle of the tangent vector) is $\epsilon\pi$ for some

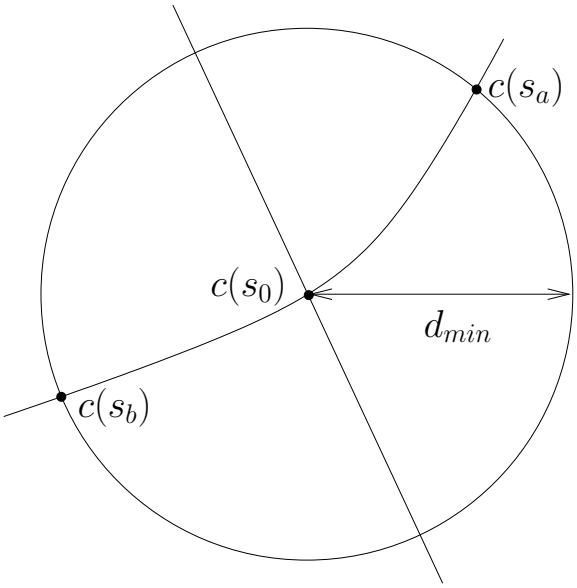

Figure 4.2: A ball of radius d_{min} centered at $c(s_0)$, partioned by the line normal to the curve at s_0.

$0 < \epsilon < 0.5$. Denote the set of endpoints of these regions as $\{s_i\}_0^n$; line segments normal to the curve at these points will be used as edges in the decomposition. This is illustrated in Figure 4.3. The following proposition will allow us to consider the discrete intervals between subsequent line segments independently.

Consider the curve c with parameterization s, and two subsequent critical points s_i and s_{i+1}. Denote the lines passing through these points and normal to the curve as n_i and n_{i+1}. Orient these lines such that for all $s \in (s_i, s_{i+1})$, $c(s)$ is in the positive halfspace of n_i and the negative halfspace of n_{i+1}, denoted h_i^+ and h_{i+1}^-, respectively. (Note that this is always possible, because of the choice of critical points described above.) Finally, continue to denote the ball of radius d_{min} centered on $c(s)$ as $B(c(s), d_{min})$.

Proposition 4.1 *Consider the tunnel of radius d_{min} centered on c between $c(s_i)$*

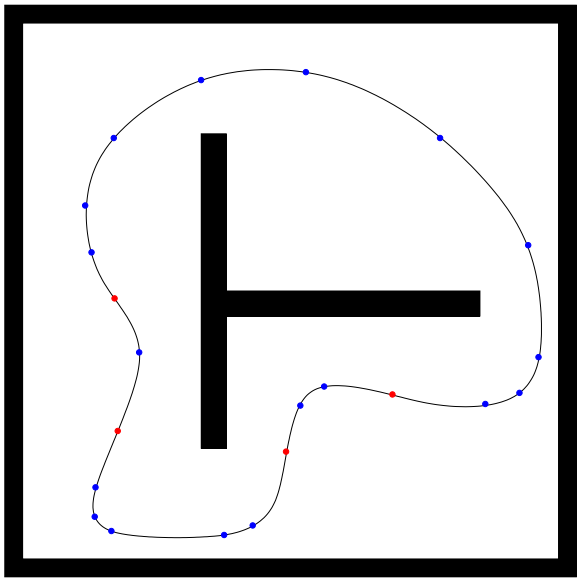

Figure 4.3: A target trajectory, divided into segments which are sufficiently simple. Red dots mark critical points due to curvature changes; blue dots mark critical points due to angular change of the trajectory's tangent vector over the interval.

and $c(s_{i+1})$*, defined as*

$$T(c, s_i, s_{i+1}, d_{min}) = \left(\bigcup_{s \in (s_i, s_{i+1})} B(c(s), d_{min}) \right) \cap h_i^+ \cap h_{i+1}^-.$$

This tunnel does not intersect with any obstacles and does not contain any $c(s)$ *for* $s \notin (s_i, s_{i+1})$*.*

Proof: First, the tunnel does not intersect with any obstacles because of the separation assumption above. Second, the tunnel does not contain any portion of the curve outside of the interval between the two critical points. Recall the curve separation assumption made above; it stated that for any point $c(s_0)$, $|c(s_1) - c(s_0)| \leq d_{min}$ implies that $|c(s) - c(s_0)| \leq d_{min}$ for all $s \in [s_0, s_1]$. This means that once the curve leaves the ball at a particular point, it cannot reenter it. As we have already seen, the curve cannot recross the normal vector to the curve at a given point without leaving the ball at that point. This implies that once the

curve crosses the line segment at the end of the interval, it cannot reenter the tunnel without leaving the ball centered at the endpoint. As a result, it is also forced to exit $B(c(s), d_{min})$ for any $s \in (s_i, s_{i+1})$, and therefore never reenters any of them, thereby never reentering the tunnel. ■

The proposition above implies that as long as a polygonal decomposition can be computed for each segment of the curve such that it lies inside the tunnel about that segment, then the polygonal cells are guaranteed to be nonoverlapping, either with the obstacles or with each other. It is unnecessary to go into significant detail on building the polygonal decomposition for each segment; many approaches are feasible, using algorithms that approximate curves with arbitrary precision. Using the fact that each curve segment has sign-invariant curvature, it is quite easy to build piecewise linear lower-approximations and upper-approximations of the curve, and shift these slightly to obtain the polygonal region between n_i and n_{i+1}, entirely inside the tunnel about that segment. For more sophisticated curve approximation approaches to use for this purpose, see [203] and the references therein.

The result of such a decomposition can be seen in Figure 4.4. Once the cells containing the trajectory have been computed, the remainder of the space can be decomposed using any convex decomposition algorithm. To ensure that the resulting decomposition respects the curve cells, treat them as "obstacles" for the purposes of the subsequent convex decomposition; then, combine the two sets of cells to obtain a decomposition of the entire free space. Figure 4.5 illustrates the final result.

Constructing the vector field

With the decomposition we have constructed, it is nearly trivial to define local vector fields that avoid obstacles and converge to the given trajectory in the cells

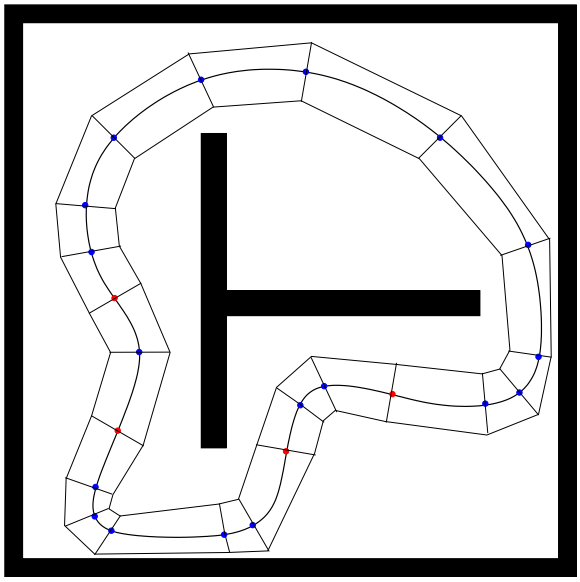

Figure 4.4: A polygonal decomposition containing the target trajectory. Individual cells are formed using line segments normal to the trajectory at the critical points shown in Figure 4.3.

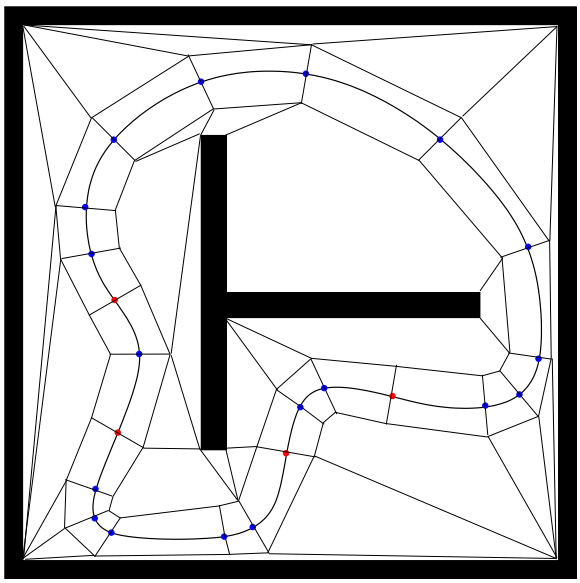

Figure 4.5: A polygonal decomposition of the entire free configuration space. The decomposition respects the decomposition in Figure 4.4.

enclosing the curve. We will define the face and cell vector fields for these cells first, and then address the cells which do not contain any curve segment. We will also initially describe the case where the trajectory is a closed curve, and then mention application to instances where it is not. As in Chapter 2, the face vector fields should prevent the robot from crossing a nonexit face while permitting it to cross the exit face (here, the exit face is clearly the face from which the curve exits). In this case, the face vector fields will also induce convergence to the trajectory. The cell vector field will cause the robot to cross the exit face of the cell (or, alternatively, cause it to progress along the trajectory).

We may build face and cell vector fields using vector fields parallel and normal to the curve in the cell. For any point p in the cell, define $r(p)$ as the point on the curve such that the line normal to the curve at $r(p)$ goes through p. This is well-defined for any point in the cells containing the curve, because they are contained within a tunnel of radius d_{min} about c, and each cell contains a sign-invariant segment of c. The curvature constraints imply that the radius of any osculating circle of c has radius greater than d_{min}; therefore, the normal lines sweep through the cells such that each point is touched only once. Therefore, we can define the perpendicular vector field as follows:

$$V_{perp}(p) = \text{norm}(r(p) - p).$$

Define the parallel vector field as

$$V_{par}(p) = T(r(p)),$$

in which $T(r(p))$ is the tangent vector to c at $r(p)$, following the convention above. We can now define the face and cell vector fields.

Consider a cell face, other than those through which the curve enters or exits. For such a face f, let V_{nf} be the inward pointing constant vector field normal to the face. Then, define the face vector field as

$$V_f(p) = \beta V_{nf} + (1 - \beta)V_{par}(p),$$

in which β is a smoothly varying parameter that equals one on the face itself and zero on the curve c. This parameter can be constructed by an analytic switch, as described in Chapter 2. This face vector field always guarantees that the vector field is decreasing the distance to the curve, as measured by $||r(p) - p||$. We know that $V_{nf} \cdot V_{perp} > 0$ because of the condition that the angular change of the curve in the cell is less than $\pi/2$; therefore, the normal vector of any face in between the entrance and exit cells is guaranteed to have positive dot product with $r(p) - p$ for any point in the cell. The parameter β ensures that as the robot approaches the target trajectory, it follows the trajectory rather than crossing over it. We have $\beta = 0$ on the face itself so that the face vector field can smoothly match its analogue in the cell on the other side of the face (which may not contain a curve segment, instead feeding into this cell through the face f). It is also possible to shape β in a variety of ways to get desired practical behavior.

For the entrance and exit faces of the cell, define $V_f = \alpha_f V_{perp} + (1 - \alpha_f)V_{par}$, for some $\alpha_f \in (0, 1)$. This vector field is smooth across the entrance and exit faces because the tangent and normal vectors of the curve change smoothly, since the curve itself is smooth. Similarly, define the cell vector field as $V_c = \alpha_c V_{perp} + (1 - \alpha_c)V_{par}$, for some $\alpha_c \in (0, 1)$. The vector field V_{par} does not increase the distance to the curve, and V_{perp} decreases it. Using the interpolation scheme described in Chapter 2, it follows almost immediately that the integral curves of this vector field converge to the specified trajectory. It is also clear that the

integral curves avoid the obstacles, because the vector field is inward-pointing on all obstacle edges. The integral curves are also smooth; the smoothness of the target trajectory means that both V_{par} and V_{perp} are smooth, so a smooth interpolation of them is also smooth.

Finally, consider two additional constructions. First, assume the trajectory is an open curve rather than a closed one. We need only to consider the initial and terminal cells; furthermore, all component vector fields except for the corresponding entrance and exit vector fields can be defined as in Chapter 2. So, consider the face vector field for the entrance face of the terminal (goal) cell; the face vector field on the other side of the face is $V_f = \alpha_f V_{perp} + (1 - \alpha_f)V_{par}$, as described above. This is suitable for the face vector field in the goal cell as well, because the vector field causes convergence toward the goal point (movement along the curve). The cell vector field for the terminal cell should be as described in Chapter 2. In the initial cell, the exit face's vector field can be defined similarly, and it is not difficult to show that a cell vector field oriented toward the exit face is sufficient to guarantee convergence.

Second, consider a target curve that intersects itself; this causes it to violate the separation requirement we made above, which allowed us to construct a safe decomposition of the environment. An example of such a trajectory is seen in the original example in Figure 4.1. We can observe that if an open ϵ-ball centered on each intersection is removed, the remainder of the curve satisfies the separation requirement. Therefore, the cells for these curve segments can be constructed. Then, construct two decompositions for the remainder of the space, one corresponding to each curve segment that was deleted when the intersection ball was removed. When the robot reaches the intersection, it follows the vector field of the decomposition constructed for the segment of the curve that it is supposed to follow, which is simple to do (recall that the vector field depends on the param-

eterization s of the curve, so the robot knows where it is along the curve). If we wish to write the vector field as a static vector field on the state space, we need to add a single new state variable which can then encode this information.

4.3 Multiple Robot Coordination

Finally, consider multiple robot coordination, which is the problem of stabilizing a set of robots to their respective goal points, while preventing the robots from colliding with obstacles or with each other. See Figure 4.6 for a simple yet challenging example of his type of problem. The problem of multiple robot coordination is well-studied; the two main types of solutions to this problem are *centralized* and *decentralized.*[1] In centralized control schemes, a central controller coordinates the actions of the individual robots to accomplish the task; this necessitates a high degree of communication between the agents. In contrast, decentralized approaches involve each robot acting on its own, with limited knowledge of each other and limited ability to communicate between them. Since we continue to assume perfect knowledge of the goal states of the robots and of the state of the system (namely, the states of each of the robots), our approach is a centralized one. It may be possible to consider decentralized versions, but almost certainly with a loss of completeness guarantees.

4.3.1 Background

Centralized controllers typically plan in the *composite* configuration (or state) space of the robots. For a set of robots $\{R_i\}$ with corresponding configuration spaces $\{\mathcal{C}_i\}$, the composite configuration space is $C = \prod_i \mathcal{C}_i$. For motion planning and control problems, this implies an enormous increase in the computational cost,

[1] In open loop multiple robot coordination work, this is typically referred to as *decoupled.*

Figure 4.6: A challenging multiple robot navigation problem. The red and blue robots must exchange places.

due to its exponential dependence on the dimension of the configuration space. Decentralized approaches compute plans for each robot independently, each in the configuration space of that individual robot. This may appear to offer hope that the problem can be solved more efficiently, since the individual configuration spaces have much lower dimension than the composite one; however, the worst case bounds cannot be bypassed so easily. Complete algorithms for planning and control still require exponential time; decentralized approaches may be more efficient in some practical cases, but the worst case bound is the same.

In addition to a great deal of work on the subject of open-loop multiple robot coordination [204–207], a number of approaches have been developed for computing closed loop plans for multiple robot systems [111, 113, 114, 208–210]. Some of the closed loop approaches are based on navigation functions, which were discussed in Chapter 2. They have the desirable properties of safety, smoothness, and completeness, but are difficult to use in practice. In contrast, the trivial extension of this approach to the case of multiple robots yields a resolution complete solution that is highly practical compared to some of these methods.

4.3.2 A polygonal approximation approach

Consider a set of disc robots $\{R_i\}_1^n$, with radii $\{r_i\}_1^n$, moving in the plane. As we described in Chapter 2, it is possible, in the case of a single robot, to compute smooth feedback plans over either the exact configuration space or its piecewise linear approximation. For our purposes here, consider the piecewise approximation of the configuration space for each robot. Without considering interrobot collisions, the composite configuration space (with $2n$ dimensions) is also clearly piecewise linear. In order to apply the methods of Chapter 2, we only need to describe the interrobot collision constraints in a piecewise linear way.

It has been widely noted in the literature that the configuration space resulting from the consideration of robot-robot collisions is *cylindrical*: for planar robots, each constraint can be specified as

$$(x_i - x_j)^2 + (y_i - y_j)^2 > (r_i + r_j)^2,$$

which is a cylinder in the four-dimensional composite configuration space. This cylinder can easily be approximated, conservatively and to an arbitrary resolution, by a set of hyperplanes (exactly as you would approximate a circle with straight line segments). We now have a composite configuration space that conservatively considers robot-obstacle and robot-robot collisions and that is piecewise linear. Therefore, the results from Chapter 2 can be directly applied, and a resolution complete solution to the closed loop multiple robot coordination problem is obtained.

While a rigorous analysis has not been carried out, we believe that further results would be straightforward to obtain. First, given the results already derived for polygonal robots in the plane, it should be straightforward to extend the approximation to the multirobot case as well. Second, it may be possible to use

the proof ideas for planar unicycles and apply them to higher dimensional unicycle models. If this is possible, then that should imply a solution for multiple unicycle disc robots as well (although unicycles with polygonal bodies pose a much more difficult problem).

4.4 Conclusion

In this chapter, several extensions of the basic approach were described. First, we saw how to construct smooth feedback plans whose integral curves converge to a target trajectory while avoiding obstacles. This will allow great flexibility in the specification of robot tasks, while providing the important guarantees of convergence, obstacle avoidance, and smoothness. Second, we outlined a simple approach for computing smooth feedback plans for the multiple robot coordination problem. Even though the problem is likely intractable for large numbers of robots (due to the dimension of the composite configuration space), it is potentially applicable for small to medium teams of robots.

CHAPTER 5

CONCLUSIONS

In this chapter, we will overview the contributions made in this dissertation, and directions for future work will be described.

5.1 Summary

The goal of this dissertation has been to present a complete, efficient solution to the global navigation problem for mobile robotics. Robots have emerged in the consumer market through products such as iRobot's Roomba vacuuming robot and Friendly Robotics' Robomow lawn mowing robot. Autonomous vehicles are already being used for many applications, such as space exploration, military operations, and traffic monitoring. Efforts like the DARPA Urban Challenge are spurring research that will eventually enable robots to drive safely on public roads, following traffic laws and avoiding collisions with pedestrians, cyclists, and other drivers. In this context, the development of algorithms that automatically guide robots to desired goal locations while avoiding obstacles is of tremendous importance. One way to approach this is to design control laws that have the properties of *safety* and *convergence*: applying the controller from any initial state guarantees that the robot will asymptotically converge to the goal state, and will not collide with any obstacles. If, in addition, the controller always yields smooth (i.e., C^∞) trajectories for the robot, the controller solves the *smooth feedback planning* problem.

In this dissertation, an approach to the smooth feedback planning problem has been presented, utilizing cell decomposition and vector field interpolation. Cell decompositions have been used for motion planning since the introduction of the configuration space by Lozano-Pérez and Wesley [9]. Decompositions have been used to exactly represent the free and obstacle regions in the configuration space, as well as to build resolution complete approximations that can be used for efficient planning. These methods gradually disappeared, as sampling-based motion planning algorithms increasingly focused on high-dimensional configuration spaces, where geometric decompositions are inefficient. The key advantage of cell decompositions in the context of smooth feedback planning is that they partition the space into pieces for which simple control laws can be constructed.

These simple control laws are constructed by smoothly interpolating between a set of vector fields defined over each cell. Many approaches to global feedback in environments with obstacles are based on potential fields, which generate control inputs by taking the gradient of the potential function. While it is easy to incorporate intuitive concepts such as "move toward the goal" and "avoid obstacles" into a potential function, it is difficult to do so in such a way that does not introduce undesirable local minima into the potential function. Rimon and Koditschek developed navigation functions to address precisely this issue; while their method is theoretically impressive, it is difficult to implement and use in practice. In contrast, the algorithms presented in this dissertation achieve both completeness and practicality.

In Chapter 2, the basic approach was presented and applied to fully actuated robots moving in polygonal and semi-algebraic environments. In polygonal environments (or, more generally, environments with piecewise-linear boundaries), construction of smooth feedback plans were constructed for PL environments of any dimension. This simple polygonal case was important in later chapters, be-

cause it provided a platform from which further results could be easily derived. A second type of cell decomposition, cylindrical algebraic decomposition, was then discussed, and it was shown how to compute smooth feedback plans over such decompositions. While not feasible for practical problems, this result solves the feedback version of the generalized piano mover's problem, which is a foundational problem in robotics. The algorithm takes a cylindrical algebraic decomposition (CAD) and generates a smooth feedback plan over that decomposition, requiring only linear time with respect to complexity of the decomposition itself. This implies that obtaining full feedback solutions to the generalized piano mover's problem is no more difficult than solving the basic motion planning problem—you get "feedback for free." Finally, decompositions were presented for state spaces arising from disc or polygon robots moving in planar polygonal environments, and algorithms for computing smooth feedback plans over these environments were presented.

In Chapter 3, smooth feedback planning algorithms were developed for simple nonholonomic robots moving in polygonal environments. Many practical robots can be modeled as simple nonholonomic systems such as the unicycle. Smooth feedback planning algorithms were presented for unicycle robots and car-like robots, which move on bounded curvature paths. The algorithms were shown to be complete, as well as practical and efficient. For unicycles, it is possible to compute smooth feedback plans with little additional effort beyond computing a plan for a fully actuated robot. The bounded curvature constraint for car-like robots implies that path reversals are necessary to traverse arbitrary environments; i.e., the robot must sometimes stop and reverse direction in order to go around sharp corners. By adding an additional state space variable (forward velocity v_f, in addition to the three configuration space variables of x, y, and θ),

it was possible to compute feedback plans in which the robot smoothly reverses direction as many times as necessary to continue toward the goal point.

In Chapter 4, we presented extensions of the basic method to several interesting problems. First, a trajectory-tracking controller was presented, whereby a robot converges to a specified smooth trajectory while avoiding obstacles. This is useful for many practical robotics applications, such as security/patrolling or for receiving guidance from a human partner. Second, the basic smooth feedback planning algorithm was applied to the problem of centralized multiple robot coordination. In this way, one can easily enable multiple robots to share an environment, each carrying out its assigned tasks without colliding with another. The primary limitation to this approach is that the dimension of the combined state space increases linearly with the number of robots, so the approach is feasible only for small teams.

5.2 Future Directions

There are a number of interesting theoretical and practical challenges that remain in this line of work. Given the promise of mobile robotics and the general efficiency of decomposition/vector field based approaches, there could be significant value in pursuing this research further.

One significant step toward general smooth feedback planning for mobile robots is the problem of nonholonomic robots with polygonal bodies. This dissertation demonstrated smooth feedback planning for disc robots, but results for polygonal robots have not been shown. While many robots are designed such that modeling them as discs is a reasonable approximation, many real world vehicles simply do not fit the bill. In many real-life problems, as well as problems in computer graphics and animation (for movies or video games, for example), autonomous vehicles

cannot be modeled as discs. What happens when a fire truck needs to cross a narrow bridge? Or school buses need to drive in parallel? Disc models simply do not suffice for these situations, yet handling these problems is imperative for comprehensively addressing the areas of mobile robotics and autonomous vehicle navigation.

A number of interesting theoretical problems remain open. For example, the results for planar unicycles can be extended to arbitrary dimensions. Applications for second order systems can be explored, perhaps in the context of spacecraft with limited thrust capabilities. It may not be possible to obtain solutions that are comprehensive and complete, but resolution complete approximations could be used to make it possible for spacecraft to perform delicate docking maneuvers under the guidance of a provably safe and convergent control law. Further integration with sampling-based motion planning algorithms can be pursued, with the methods of this dissertation being used to create local control policies without the cost of performing a decomposition of the entire configuration or state space.

These methods are practical and useful for real robotics applications, in addition to being theoretically sound. The algorithms presented here can be used in conjunction with complex decision logic to solve a variety of behavior tasks. The methods sketched in this dissertation, for path following and multiple robot coordination, have many applications. The greatest amount of work, practically speaking, could easily lie in developing and implementing practical convex decomposition algorithms for high dimensional spaces. Vertical decomposition can easily handle this challenge, but we know of no actual implementation that can robustly handle piecewise linear spaces in arbitary dimensions. For more on this problem in the context of computational geometry, see Halperin [174]. In contrast, there are a number of outstanding convex decomposition algorithm implementations for planar environments, such as Triangle [211]. The ability to perform convex de-

compositions in higher dimensions might make centralized approaches to multiple robot coordination problems feasible for small teams of robots (perhaps with 3-5 members).

Robotics and automation will be pivotal in advances in science and technology over the coming decades. The changes that will most impact people's lives involve robots in the real world—moving and interacting with human beings in human-centered environments. Moreover, robots will need to operate robustly, without the need for human help every time an unexpected situation arises. Feedback control can address some of these challenges, but the need for obstacle avoidance must be included from the outset. This dissertation has presented first steps at what may prove to be a profitable approach for robust real-world robot navigation.

REFERENCES

[1] iRobot Corporation, "iRobot at a glance," 2008. [Online]. Available: http://www.irobot.com/sp.cfm?pageid=203.

[2] BBC NEWS, "US plans 'robot troops' for Iraq," 1995. [Online]. Available: http://news.bbc.co.uk/2/hi/americas/4199935.stm.

[3] P. Lerner, "The army's robot sherpa," *Popular Science*, 2006. [Online]. Available: http://popsci.com/scitech/article/2006-03/armys-robot-sherpa.

[4] W. Knight, "Beer-bot pours chilled drinks for thirsty humans," *New Scientist*, 2006. [Online]. Available: http://www.newscientist.com/article.ns?id=dn8642.

[5] N. Roy, G. Baltus, D. Fox, F. Gemperle, J. Goetz, T. Hirsch, D. Margaritis, M. Montemerlo, J. Pineau, J. Schulte, and S. Thrun, "Towards personal service robots for the elderly," in *Proceedings Workshop in Interactive Robots and Entertainment*, 2000.

[6] W. Gates, "A robot in every home," *Scientific American*, pp. 938–948, Jan. 2007.

[7] S. Cherry, "Robots, incorporated," *IEEE Spectrum*, pp. 24–29, Aug. 2007.

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

[9] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Communications of the ACM*, vol. 22, no. 10, pp. 560–570, 1979.

[10] T. Lozano-Pérez, "Spatial planning: A configuration space approach," *IEEE Transactions on Computing*, vol. C-32, no. 2, pp. 108–120, 1983.

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

[12] J. T. Schwartz and M. Sharir, "On the piano movers' problem: II. General techniques for computing topological properties of algebraic manifolds," *Communications on Pure and Applied Mathematics*, vol. 36, pp. 345–398, 1983.

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

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

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

[16] R. A. Brooks and T. Lozano-Pérez, "A subdivision algorithm in configuration space for findpath with rotation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-15, no. 2, pp. 224–233, 1985.

[17] B. R. Donald, "Motion planning with six degrees of freedom," Artificial Intelligence Lab., Massachusetts Institute of Technology, Cambridge, MA, Tech. Rep. AI-TR-791, 1984.

[18] B. R. Donald, "A search algorithm for motion planning with six degrees of freedom," *Artificial Intelligence Journal*, vol. 31, pp. 295–353, 1987.

[19] T. Lozano-Pérez, "A simple motion-planning algorithm for general robot manipulators," *IEEE Journal of Robotics and Automation*, vol. RA-3, no. 3, pp. 224–238, June 1987.

[20] Y. K. Hwang and N. Ahuja, "A potential field approach to path planning," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 1, pp. 23–32, Feb. 1992.

[21] B. Faverjon and P. Tournassoud, "A local based method for path planning of manipulators with a high number of degrees of freedom," in *Proceedings IEEE International Conference on Robotics and Automation*, 1987, pp. 1152–1159.

[22] B. Faverjon, "Hierarchical object models for efficient anti-collision algorithms," in *Proceedings IEEE International Conference on Robotics and Automation*, 1989, pp. 333–340.

[23] B. Paden, A. Mees, and M. Fisher, "Path planning using a Jacobian-based freespace generation algorithm," in *Proceedings IEEE International Conference on Robotics and Automation*, 1989, pp. 1732–1737.

[24] P. C. Chen and Y. K. Hwang, "SANDROS: A motion planner with performance proportional to task difficulty," in *Proceedings IEEE International Conference on Robotics and Automation*, 1992, pp. 2346–2353.

[25] K. Kondo, "Motion planning with six degrees of freedom by multistrategic bidirectional heuristic free-space enumeration," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 267–277, 1991.

[26] J. Lengyel, M. Reichert, B. R. Donald, and D. P. Greenberg, "Real-time robot motion planning using rasterizing computer graphics hardware," *Computer Graphics*, vol. 24, no. 4, pp. 327–335, Aug. 1990.

[27] R. Bohlin, "Path planning in practice; lazy evaluation on a multi-resolution grid," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, pp. 49–54.

[28] R. Bohlin and L. Kavraki, "Path planning using Lazy PRM," in *Proceedings IEEE International Conference on Robotics and Automation*, 2000, pp. 521–528.

[29] G. Sánchez and J.-C. Latombe, "A single-query bi-directional probabilistic roadmap planner with lazy collision checking," in *Proceedings International Symposium on Robotics Research*, 2001, pp. 403–417.

[30] J. Barraquand and J.-C. Latombe, "A Monte-Carlo algorithm for path planning with many degrees of freedom," in *Proceedings IEEE International Conference on Robotics and Automation*, 1990, pp. 1712–1717.

[31] 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*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[32] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State University, Tech. Rep. 98-11, Oct. 1998.

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

[34] E. Mazer, J. M. Ahuactzin, and P. Bessière, "The Ariadne's clew algorithm," *Journal of Artificial Intelligence Research*, vol. 9, pp. 295–316, Nov. 1998.

[35] E. Mazer, G. Talbi, J. M. Ahuactzin, and P. Bessière, "The Ariadne's clew algorithm," in *Proceedings International Conference of Society of Adaptive Behavior*, Honolulu, 1992, pp. 182–188.

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

[37] 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*, Zeist, The Netherlands, 2004, pp. 297–312.

[38] F. Lingelbach, "Path planning using probabilistic cell decomposition," in *Proceedings IEEE International Conference on Robotics and Automation*, 2004, pp. 467–472.

[39] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "Choosing good distance metrics and local planners for probabilistic roadmap methods," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 4, pp. 442–447, Aug. 2000.

[40] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *Proceedings Workshop on Algorithmic Foundations of Robotics*, 1998, pp. 155–168.

[41] B. Burns and O. Brock, "Sampling-based motion planning using predictive models," in *Proceedings IEEE International Conference on Robotics and Automation*, 2005, pp. 3120–3125.

[42] B. Burns and O. Brock, "Toward optimal configuration space sampling," in *Proceedings Robotics: Science and Systems*, 2005, pp. 105–112.

[43] P. Leven and S. A. Hutchinson, "Real-time path planning in changing environments," *IEEE Transactions on Robotics and Automation*, vol. 21, no. 12, pp. 999–1030, Dec. 2002.

[44] C. Pisula, K. Hoff, M. Lin, and D. Manocha, "Randomized path planning for a rigid body based on hardware accelerated Voronoi sampling," in *Proceedings Workshop on Algorithmic Foundations of Robotics*, 2000, pp. 279–292.

[45] E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki, "Sampling-based roadmap of trees for parallel motion planning," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 597–608, 2005.

[46] T. Siméon, J.-P. Laumond, and C. Nissoux, "Visibility based probabilistic roadmaps for motion planning," *Advanced Robotics*, vol. 14, no. 6, pp. 477–493, 2000.

[47] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *Proceedings IEEE International Conference on Robotics and Automation*, 1999, pp. 1024–1031.

[48] J. Yakey, S. M. LaValle, and L. E. Kavraki, "Randomized path planning for linkages with closed kinematic chains," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 951–958, Dec. 2001.

[49] Y. Koga, K. Kondo, J. Kuffner, and J.-C. Latombe, "Planning motions with intentions," *Proceedings ACM SIGGRAPH*, pp. 395–408, 1994.

[50] J. Pettré, J.-P. Laumond, and T. Siméon, "A 2-stages locomotion planner for digital actors," in *Proceedings Eurographics/SIGGRAPH Symposium on Computer Animation*, 2003, pp. 258–264.

[51] G. Song and N. M. Amato, "Using motion planning to study protein folding pathways," *Journal of Computational Biology*, vol. 26, no. 2, pp. 282–304, 2002.

[52] A. Bhatia and E. Frazzoli, "Incremental search methods for reachability analysis of continuous and hybrid systems," in *Hybrid Systems: Computation and Control*, R. Alur and G. J. Pappas, Eds. Berlin: Springer-Verlag, 2004, pp. 67–78.

[53] M. S. Branicky, M. M. Curtiss, J. Levine, and S. Morgan, "RRTs for nonlinear, discrete, and hybrid planning and control," in *Proceedings IEEE Conference Decision and Control*, 2003, pp. 657–663.

[54] J. Bruce and M. Veloso, "Real-time randomized path planning for robot navigation," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002, pp. 2383–2388.

[55] P. Cheng and S. M. LaValle, "Resolution complete rapidly-exploring random trees," in *Proceedings IEEE International Conference on Robotics and Automation*, 2002, pp. 267–272.

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

[57] M. J. de Smith, "Distance and path: The development, interpretation and application of distance measurement in mapping and modelling," Ph.D. dissertation, University College, University of London, London, 2003.

[58] D. Ferguson and A. Stentz, "Replanning with RRTs," in *Proceedings IEEE International Conference on Robotics and Automation*, 2006, pp. 1243–1248.

[59] E. Frazzoli, M. A. Dahleh, and E. Feron, "Real-time motion planning for agile autonomous vehicles," *AIAA Journal of Guidance and Control*, vol. 25, no. 1, pp. 116–129, 2002.

[60] S. Kagami, J. Kuffner, K. Nishiwaki, and K. O. M. Inaba, "Humanoid arm motion planning using stereo vision and RRT search," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003, pp. 2167–2172.

[61] M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann, "Planning collision-free reaching motions for interactive object manipulation and grasping," *Eurographics*, vol. 22, no. 3, pp. 313–322, 2003.

[62] J. Kim and J. P. Ostrowski, "Motion planning a aerial robot using rapidly-exploring random trees with dynamic constraints," in *Proceedings IEEE International Conference on Robotics and Automation*, 2003, pp. 2200–2205.

[63] S. R. Lindemann and S. M. LaValle, "Incrementally reducing dispersion by increasing Voronoi bias in RRTs," in *Proceedings IEEE International Conference on Robotics and Automation*, 2004, pp. 3251–3257.

[64] S. R. Lindemann and S. M. LaValle, "Steps toward derandomizing RRTs," in *IEEE Fourth International Workshop on Robot Motion and Control*, 2004, pp. 271–277.

[65] T.-Y. Li and Y.-C. Shie, "An incremental learning approach to motion planning with roadmap management," in *Proceedings IEEE International Conference on Robotics and Automation*, 2002, pp. 3411–3416.

[66] S. Rodriguez, X. Tang, J.-M. Lien, and N. M. Amato, "OBRRT: An obstacle-based rapidly-exploring random tree," in *Proceedings IEEE International Conference on Robotics and Automation*, 2006, pp. 895–900.

[67] C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003, pp. 1178–1183.

[68] A. Yershova, L. Jaillet, T. Simeon, and S. M. LaValle, "Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain," in *Proceedings IEEE International Conference on Robotics and Automation*, 2005, pp. 3856–3861.

[69] J.-P. Laumond, "Trajectories for mobile robots with kinematic and environment constraints," in *Proceedings International Conference on Intelligent Autonomous Systems*, 1986, pp. 346–354.

[70] J. Barraquand and J.-C. Latombe, "Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles," *Algorithmica*, vol. 10, pp. 121–155, 1993.

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

[72] J.-P. Laumond, S. Sekhavat, and F. Lamiraux, "Guidelines in nonholonomic motion planning for mobile robots," in *Robot Motion Planning and Control*, J.-P. Laumond, Ed. Berlin: Springer-Verlag, 1998, pp. 1–53.

[73] S. Sekhavat, P. Svestka, J.-P. Laumond, and M. H. Overmars, "Multilevel path planning for nonholonomic robots using semiholonomic subsystems," *International Journal of Robotics Research*, vol. 17, pp. 840–857, 1998.

[74] F. Bullo and K. M. Lynch, "Kinematic controllability for decoupled trajectory planning in underactuated mechanical systems," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 4, pp. 402–412, 2001.

[75] P. Cheng, "Sampling-based motion planning with differential constraints," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Aug. 2005.

[76] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[77] S. R. Lindemann and S. M. LaValle, "A multiresolution approach for motion planning under differential constraints," in *Proceedings IEEE International Conference on Robotics and Automation*, 2006, pp. 139–144.

[78] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1077–1091, Dec. 2005.

[79] W. L. Brogan, *Modern Control Theory*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 1991.

[80] C.-T. Chen, *Linear System Theory and Design*, 3rd ed. New York: Oxford University Press, 1999.

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

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

[83] 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*. Cambridge, MA: MIT Press, 2005.

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

[85] J. Barraquand and J.-C. Latombe, "Robot motion planning: A distributed representation approach," *International Journal of Robotics Research*, vol. 10, no. 6, pp. 628–649, Dec. 1991.

[86] J. E. Bobrow, S. Dubowsky, and J. S. Gibson, "Time-optimal control of robotic manipulators along specified paths," *International Journal of Robotics Research*, vol. 4, no. 3, pp. 3–17, 1985.

[87] K. G. Shin and N. D. McKay, "Minimum-time control of robot manipulators with geometric path constraints," *IEEE Transactions on Automatic Control*, vol. 30, no. 6, pp. 531–541, 1985.

[88] F. Lamiraux and J.-P. Laumond, "Flatness and small-time controllability of multibody mobile robots: Application to motion planning," *IEEE Transactions on Automatic Control*, vol. 45, no. 10, pp. 1878–1881, Oct. 2000.

[89] R. M. Murray and S. Sastry, "Nonholonomic motion planning: Steering using sinusoids," *IEEE Transactions on Automatic Control*, vol. 38, no. 5, pp. 700–716, 1993.

[90] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.

[91] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, pp. 407–427, 1999.

[92] 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*, vol. 25, no. 4, pp. 755–764, 2002.

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

[94] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[95] B. H. Krogh, "A generalized potential field approach to obstacle avoidance control," in *Proceedings of SME Conference on Robotics Research*, Aug. 1984, pp. 14–19.

[96] R. Volpe and P. Khosla, "Manipulator control with superquadric artificial potential functions: Theory and experiments," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 6, pp. 1423–1436, 1990.

[97] T. Hebert, "Navigation of an autonomous vehicle using a combined electrostatic potential field/fuzzy inference approach," Ph.D. dissertation, Univ. Southwestern Louisiana, 1998.

[98] 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*, vol. 30, no. 3, pp. 251–272, May 2000.

[99] Y. Zhang, "Sensor-based potential field method for robot motion planning," Ph.D. dissertation, Univ. Southwestern Louisiana, 1995.

[100] C. I. Connolly, "Applications of harmonic functions to robotics," in *IEEE Symposium on Intelligent Control*, 1992, pp. 498–502.

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

[102] C. Connolly and R. Grupen, "The application of harmonic potential functions to robotics," *Journal of Robotic Systems*, vol. 10, no. 7, pp. 931–946, 1993.

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

[104] M. R. Stan, W. P. Burleson, C. I. Connolly, and R. A. Grupen, "Analog VLSI for robot path planning," *Journal of VLSI Signal Processing*, vol. 8, no. 1, pp. 61–73, 1994.

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

[106] 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*, vol. 30, no. 2, pp. 187–196, March 2000.

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

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

[109] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential fields," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, Oct. 1992.

[110] 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, pp. 3012–3017.

[111] S. G. Loizou and K. J. Kyriakopoulos, "Closed loop navigation of multiple holonomic vehicles," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002, pp. 2861–2866.

[112] S. Loizou and K. Kyriakopoulos, "Closed loop navigation for multiple non-holonomic vehicles," in *Proceedings IEEE International Conference on Robotics and Automation*, 2003, pp. 420–425.

[113] S. G. Loizou and K. J. Kyriakopoulos, "Navigation of multiple kinematically constrained robots," *IEEE Transactions on Robotics and Automation*, vol. 24, no. 1, pp. 221–231, Feb. 2008.

[114] H. G. Tanner, S. G. Loizou, and K. J. Kyriakopoulos, "Nonholonomic navigation and control of cooperating mobile manipulators," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 1, pp. 53–64, Feb. 2003.

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

[116] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.

[117] J. Borenstein and Y. Koren, "The vector field histogram – fast obstacle avoidance for mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, 1991.

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

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

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

[121] 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*, 1998, pp. 1615–1621.

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

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

[124] P. Ogren and N. E. Leonard, "A convergent dynamic window approach to obstacle avoidance," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 188–195, April 2005.

[125] 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*, 2002, pp. 508–513.

[126] 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*, vol. 20, no. 3, pp. 419–432, June 2004.

[127] R. Bohlin, "Robot path planning," Ph.D. dissertation, Chalmers University, Gothenburg, Sweden, 2002.

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

[129] P. Y. Li and R. Horowitz, "Passive velocity field control of mechanical manipulators," *IEEE Transactions on Robotics and Automation*, vol. 15, no. 4, pp. 751–763, Aug. 1999.

[130] P. Y. Li and R. Horowitz, "Passive velocity field control (PVFC): Part I–Geometry and robustness," *IEEE Transactions on Automatic Control*, vol. 46, no. 9, pp. 1346–1359, Sep. 2001.

[131] P. Y. Li and R. Horowitz, "Passive velocity field control (PVFC): Part II–Application to contour following," *IEEE Transactions on Automatic Control*, vol. 46, no. 9, pp. 1360–1371, Sep. 2001.

[132] I. Cervantes, R. Kelly, J. Alvarez-Ramirez, and J. Moreno, "A robust velocity field control," *IEEE Transactions on Control Systems Technology*, vol. 10, no. 6, pp. 888–894, Nov. 2002.

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

[134] J. Moreno-Valenzuela, "A new velocity field controller for robot arms," in *Proceedings IEEE International Conference on Robotics and Automation*, 2002, pp. 13–18.

[135] 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*, 1998, pp. 368–373.

[136] 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*, 1997, pp. 11–16.

[137] 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*, 2005, pp. 145–150.

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

[139] S. M. LaValle and P. Konkimalla, "Algorithms for computing numerical optimal feedback motion strategies," *International Journal of Robotics Research*, vol. 20, no. 9, pp. 729–752, Sep. 2001.

[140] I. M. Mitchell and S. Sastry, "Continuous path planning with multiple constraints," in *Proceedings IEEE Conference on Decision and Control*, 2003, pp. 5502–5507.

[141] J. A. Sethian, "A fast marching level set method for monotonically advancing fronts," *Proceedings of the National Academy of Sciences*, vol. 93, no. 4, pp. 1591–1595, 1996.

[142] J. A. Sethian and A. Vladimirsky, "Ordered upwind methods for static Hamilton-Jacobi equations: Theory and algorithms," *SIAM Journal on Numerical Analysis*, vol. 41, no. 1, pp. 325–363, 2003.

[143] J. N. Tsitsiklis, "Efficient algorithms for globally optimal trajectories," *IEEE Transactions on Automatic Control*, vol. 40, no. 9, pp. 1528–1538, Sep. 1995.

[144] 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*, vol. 138, no. 1, pp. 3–34, 1995.

[145] M. Branicky, "Studies in hybrid systems: Modeling, analysis, and control," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1995.

[146] M. Branicky, "A unified framework for hybrid control: Model and optimal control theory," *IEEE Transactions on Automatic Control*, vol. 43, no. 1, pp. 31–45, Jan. 1998.

[147] M. Branicky, "Multiple Lyapunov functions and other analysis tools for switched and hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 475–482, April 1998.

[148] 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*, vol. 43, no. 1, pp. 31–45, Jan. 1998.

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

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

[151] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *International Journal of Robotics Research*, vol. 18, no. 6, pp. 534–555, 1999.

[152] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *International Journal of Robotics Research*, vol. 3, no. 1, pp. 3–24, 1984.

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

[154] F. Blanchini and S. Miani, "A new class of universal Lyapunov functions for the control of uncertain linear systems," *IEEE Transactions on Automatic Control*, vol. 44, no. 3, pp. 641–647, March 1999.

[155] M. W. McConley, M. A. Dahleh, and E. Feron, "Polytopic control Lyapunov functions for robust stabilization of a class of nonlinear systems," in *Proceedings IEEE American Control Conference*, 1997, pp. 416–419.

[156] Y. Ohta, "On the construction of piecewise linear Lyapunov functions," in *Proceedings IEEE Conference on Decision and Control*, 2001, pp. 2173–2178.

[157] Y. Ohta, H. Imanishi, L. Gong, and H. Haneda, "Computer generated Lyapunov functions for a class of nonlinear systems," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 40, no. 5, pp. 343–354, May 1993.

[158] Y. Ohta and M. Tsuji, "A generalization of piecewise linear Lyapunov functions," in *Proceedings IEEE Conference on Decision and Control*, 2003, pp. 5091–5096.

[159] C. Belta, V. Isler, and G. J. Pappas, "Discrete abstractions for robot motion planning and control in polygonal environments," *IEEE Transactions on Robotics*, vol. 21, no. 5, pp. 864–874, Oct. 2005.

[160] 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*, vol. 51, no. 6, pp. 938–948, June 2006.

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

[162] 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*, 2005, pp. 4706–4711.

[163] 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*, 2002, pp. 534–539.

[164] M. Kloetzer and C. Belta, "Reachability analysis of multi-affine systems," in *Hybrid Systems: Computation and Control*, J. Hespanha and A. Tiwari, Eds. Berlin: Springer-Verlag, 2006, pp. 348–362.

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

[166] 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*, 2005, pp. 4175–4180.

[167] 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*, 2005, pp. 4885–4890.

[168] D. C. Conner, A. A. Rizzi, and H. Choset, "Composition of local potential functions for global robot control and navigation," in *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003, pp. 3546–3551.

[169] 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*, 2006, pp. 57–64.

[170] S. R. Lindemann and S. M. LaValle, "Smoothly blending vector fields for global robot navigation," in *Proceedings IEEE Conference Decision and Control*, 2005, pp. 3353–3559.

[171] 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 and Control*, 2006, pp. 2406–2411.

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

[173] R. Seidel, "A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons," *Computational Geometry: Theory and Applications*, vol. 1, no. 1, pp. 51–64, 1991.

[174] D. Halperin, "Arrangements," in *Handbook of Discrete and Computational Geometry*, 2nd ed., J. E. Goodman and J. O'Rourke, Eds. New York: Chapman and Hall/CRC Press, 2004, pp. 529–562.

[175] P. F. Rowat, "Representing the spatial experience and solving spatial problems in a simulated robot environment," Ph.D. dissertation, University of British Columbia, 1979.

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

[177] D. G. Kirkpatrick, "Optimal search in planar subdivisions," *SIAM Journal of Computing*, vol. 12, pp. 28–35, 1983.

[178] J. E. Goodman and J. O'Rourke, Eds., *Handbook of Discrete and Computational Geometry*, 2nd ed. New York: Chapman and Hall/CRC Press, 2004.

[179] B. Chazelle, "Triangulating a simple polygon in linear time," *Discrete and Computational Geometry*, vol. 6, no. 5, pp. 485–524, 1991.

[180] A. Narkhede and D. Manocha, "Fast polygon triangulation based on seidel's algorithm," in *Graphics Gems V*, A. Paeth, Ed. New York: Academic, 1995, pp. 394–397.

[181] J. Bañon, "Implementation and extension of the ladder algorithm," in *Proceedings IEEE International Conference on Robotics and Automation*, 1990, pp. 1548–1553.

[182] 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*, 1988, pp. 1656–1660.

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

[184] B. Mishra, "Computational real algebraic geometry," in *Handbook of Discrete and Computational Geometry*, J. E. Goodman and J. O'Rourke, Eds. New York: CRC Press, 1997, pp. 537–556.

[185] A. Strzebonski, "Cylindrical algebraic decomposition using validated numerics," *Journal of Symbolic Computation*, vol. 41, pp. 1021–1038, 2006.

[186] A. Bloch, J. Baillieul, P. Crouch, and J. Marsden, *Nonholonomic Mechanics and Control*. New York: Springer-Verlag, 2003.

[187] F. Bullo and A. D. Lewis, *Geometric Control of Mechanical Systems*. Berlin: Springer-Verlag, 2004.

[188] J. Cortés, S. Martínez, J. P. Ostrowski, and H. Zhang, "Simple mechanical control systems with constraints and symmetry," *SIAM Journal on Control and Optimization*, vol. 41, no. 3, pp. 851–874, 2002.

[189] I. Hussein and A. Bloch, "Optimal control of underactuated nonholonomic mechanical systems," in *Proceedings American Control Conference*, 2006, pp. 5590–5595.

[190] C. Samson and K. Ait-Abderrahim, "Feedback control of a nonholonomic wheeled cart in Cartesian space," in *Proceedings IEEE International Conference on Robotics and Automation*, 1991, pp. 1136–1141.

[191] C. Samson, "Time-varying feedback stabilization of car-like wheeled mobile robots," *International Journal of Robotics Research*, vol. 12, pp. 55–65, 1993.

[192] A. D. Luca, G. Oriolo, and C. Samson, "Feedback control of a nonholonomic car-like robot," in *Robot Motion Planning and Control*, J.-P. Laumond, Ed. Berlin: Springer-Verlag, 1998, pp. 171–253.

[193] M. Egerstedt, X. Hu, and A. Stotsky, "Control of a car-like robot using a virtual vehicle approach," in *Proceedings IEEE Conference on Decision and Control*, 1998, pp. 1502–1507.

[194] A. Jadbabaie, "Nonlinear receding horizon control: A control Lyapunov function approach," Ph.D. dissertation, California Institute of Technology, Pasadena, CA, 2001.

[195] A. Jadbabaie, J. Yu, and J. Hauser, "Unconstrained receding horizon control of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 46, no. 5, pp. 776–783, 2001.

[196] J. A. Primbs, V. Nevistic, and J. C. Doyle, "Nonlinear optimal control: A control Lyapunov approach," *Asian Journal of Control*, vol. 1, no. 1, pp. 14–24, March 1999.

[197] Y. Kuwata, A. Richards, T. Schouwenaars, and J. P. How, "Decentralized robust receding horizon control for multi-vehicle guidance," in *Proceedings IEEE American Control Conference*, 2006, pp. 2047–2052.

[198] H. G. Tanner, S. Loizou, and K. J. Kyriakopoulos, "Nonholonomic stabilization and collision avoidance for mobile robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2001, pp. 1220–1225.

[199] R. A. Brooks, "Solving the find-path problem by good representation of free space," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, no. 3, pp. 190–197, 1983.

[200] N. Sarkar, X. Yun, and V. Kumar, "Dynamic path following: A new control algorithm for mobile robots," in *Proceedings IEEE Conference on Decision and Control*, 1993, pp. 2670–2675.

[201] M. Egerstedt, X. Hu, and A. Stotsky, "Control of mobile platforms using a virtual vehicle approach," *IEEE Transactions on Automatic Control*, vol. 46, no. 11, pp. 1777–1782, Nov. 2001.

[202] J. Moreno-Valenzuela, "Tracking control of on-line time-scaled trajectories for robot manipulators under constrained torques," in *Proceedings IEEE International Conference on Robotics and Automation*, 2006, pp. 19–24.

[203] J. A. Horst and I. Beichel, "A simple algorithm for efficient piecewise linear approximation of space curves," in *Proceedings International Conference on Image Processing*, vol. 2, 1997, pp. 744–747.

[204] S. Akella and S. Hutchinson, "Coordinating the motions of multiple robots with specified trajectories," in *Proceedings IEEE International Conference on Robotics and Automation*, 2002, pp. 624–631.

[205] J. Peng and S. Akella, "Coordinating multiple robots with kinodynamic constraints along specified paths," in *Algorithmic Foundations of Robotics V (WAFR 2002)*, J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson, Eds. Berlin: Springer-Verlag, 2002, pp. 221–237.

[206] T. Siméon, S. Leroy, and J.-P. Laumond, "Path coordination for multiple mobile robots: A resolution complete algorithm," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 1, pp. 42–49, Feb. 2002.

[207] R. Ghrist, J. M. O'Kane, and S. M. LaValle, "Computing pareto optimal coordinations on roadmaps," *The International Journal of Robotics Research*, vol. 24, no. 11, pp. 997–1010, 2005.

[208] C. Karagoz, H. Bozma, and D. Koditschek, "On the coordinated navigation of multiple independent disk-shaped robots," University of Pennsylvania, Philadelphia, PA, Tech. Rep. MS-CIS-07-16, Jan. 2003.

[209] L. Whitcomb and D. Koditschek, "Toward the automatic control of robot assembly tasks via potential functions: The case of 2-D sphere assemblies," in *Proceedings IEEE International Conference on Robotics and Automation*, 1992, pp. 2186–2191.

[210] L. Whitcomb and D. Koditschek, "Automatic assembly planning and control via potential functions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1991, pp. 17–23.

[211] J. R. Shewchuk, "Triangle: Engineering a 2D quality mesh generator and Delaunay triangulator," in *Applied Computational Geometry: Towards Geometric Engineering*, M. C. Lin and D. Manocha, Eds. London: Springer-Verlag, 1996, vol. 1148, pp. 203–222.

CURRICULUM VITAE

RESEARCH INTERESTS

Robotics, control theory, algorithmic motion planning, nonholonomic systems, autonomous vehicles, computational geometry, computer animation.

EDUCATION

University of Illinois at Urbana-Champaign, Urbana, IL

Ph.D. Candidate, Electrical and Computer Engineering (expected graduation date: December 2008)

- Dissertation Title: "Smooth Feedback Planning"
- Adviser: Steven M. LaValle

M.S., Electrical and Computer Engineering, May 2005

- Thesis Title: "Optimal Incremental Grid Sampling"
- Adviser: Steven M. LaValle

University of Kentucky, Lexington, KY

B.S., Electrical Engineering (Minor: Mathematics), May 2001

Covenant College, Lookout Mountain, GA

B.A., Natural Sciences, May 2000

HONORS AND AWARDS

Harriet and Robert Perry Fellowship Award, 2006-2007

National Science Foundation Graduate Research Fellowship, 2002-2006

ECE Distinguished Fellowship, 2001-2004

University Fellowship, 2001-2002

University of Kentucky: graduated summa cum laude, 2001

Covenant College: graduated summa cum laude, 2000

ACADEMIC EXPERIENCE

University of Illinois at Urbana-Champaign, Urbana, IL

Research Assistant/Fellow, August 2001 - present

- Conducted Ph.D. and M.S. research and coursework.
- Designed and implemented algorithms in C++, Java, and MATLAB.
- Contributed to open source project, the Motion Strategy Library.

Teaching Assistant, June-August 2004

ECE 110, Introduction to Electrical and Computer Engineering, Summer II 2004

- Primary instructor for introductory electrical engineering course.
- Developed lectures and examinations, assigned grades.

JOURNAL PUBLICATIONS

S. M. LaValle, M. S. Branicky, and S. R. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *International Journal of Robotics*

Research, 23(7/8):673–692, July/August 2004.

JOURNAL ARTICLES UNDER REVIEW

S. R. Lindemann and S. M. LaValle. Simple and Efficient Algorithms for Computing Smooth, Collision-Free Feedback Laws. Submitted to *International Journal of Robotics Research*, 2008.

REFEREED CONFERENCE PUBLICATIONS

S. R. Lindemann and S. M. LaValle. Smooth feedback for car-like vehicles in polygonal environments. In *Proceedings IEEE International Conference on Robotics & Automation*, 2007.

S. R. Lindemann, I. I. Hussein, and S. M. LaValle. Real time feedback control for nonholonomic mobile robots with obstacles. In *Proceedings IEEE Conference on Decision & Control*, 2006.

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

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

S. R. Lindemann and P. Cheng. Iteratively locating Voronoi vertices for dispersion estimation. In *Proceedings IEEE International Conference on Robotics and Automation*, 2005.

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.

S. R. Lindemann and S. M. LaValle. Incrementally reducing dispersion by increasing Voronoi bias in RRTs. In *Proceedings IEEE International Conference on Robotics and Automation*, 2004.

S. R. Lindemann and S. M. LaValle. Steps toward derandomizing RRTs. In *IEEE Fourth International Workshop on Robot Motion and Control*, 2004.

S. R. Lindemann, A. Yershova, and S. M. LaValle. Incremental grid sampling strategies in robotics. In *Proceedings Workshop on Algorithmic Foundations of Robotics*, pages 297–312, 2004.

S. R. Lindemann and S. M. LaValle. Current issues in sampling-based motion planning. In P. Dario and R. Chatila, editors, *Proceedings International Symposium on Robotics Research*. Springer-Verlag, Berlin, 2003.

S. R. Lindemann and S. M. LaValle. Incremental low-discrepancy lattice methods for motion planning. In *Proceedings IEEE International Conference on Robotics and Automation*, pages 2920–2927, 2003.

PRESENTATIONS

Paper presentation, "Smooth feedback for car-like vehicles in polygonal environments," at the IEEE Conference on Robotics and Automation. Rome, Italy, April 2007.

Paper presentation, "Real time feedback control for nonholonomic mobile robots with obstacles," at the IEEE Conference on Decision & Control. San Diego, CA, December 2006.

Paper presentation, "A multiresolution approach for motion planning under differential constraints," at the IEEE International Conference on Robotics and Automation. Orlando, FL, April 2006.

Paper presentation, "Iteratively locating Voronoi vertices for dispersion estimation," at the IEEE International Conference on Robotics and Automation. Barcelona, Spain, April 2005.

Paper presentation, "Smoothly blending vector fields for global robot navigation," at the IEEE Conference on Decision & Control. Seville, Spain, December 2005.

Paper presentation, "Incrementally reducing dispersion by increasing Voronoi bias in RRTs," at the IEEE International Conference on Robotics and Automation. New Orleans, LA, April 2004.

Paper presentation, "Incremental grid sampling strategies in robotics," at the Workshop on the Algorithmic Foundations of Robotics, Zeist, Netherlands, July 2004.

Invited talk, "Directions in Derandomizing Motion Planning, " at the University of Pennsylvania. Philadelphia, PA, February 2004.

Paper presentation, "Incremental low-discrepancy lattice methods for motion planning," at the IEEE International Conference on Robotics and Automation. Taipei, Taiwan, September 2003.