Real Time Feedback Control for Nonholonomic Mobile Robots With Obstacles

Stephen R. Lindemann, Islam I. Hussein and Steven M. LaValle

Abstract— **We introduce a method for constructing smooth feedback laws for a nonholonomic robot in a 2-dimensional polygonal workspace. First, we compute a smooth feedback law in the workspace without taking the nonholonomic constraints into account. We then give a general technique for using this to construct a new smooth feedback law over the entire 3-dimensional configuration space (consisting of position and orientation). The trajectories of the resulting feedback law will be smooth and will stabilize the position of the robot in the plane, neglecting the orientation. Our method is suitable for real time implementation and can be applied to dynamic environments.**

I. Introduction

Finding smooth feedback laws for global robot navigation is a long-standing problem in mobile robotics. Traditional feedback control methods fail due to non-convex constraints induced by obstacles in the environment. Algorithms developed in the motion planning community are extremely effective at computing open-loop plans in the midst of non-convex constraints [13], but generally cannot provide closed-loop control. Some have tried to address this problem by constructing potential functions, the gradient of which can be used as the feedback plan. If the potential function has no local minima other than the global minimum at the goal state, then the feedback plan solves the global navigation problem [10], [19]. Our approach is more direct. Instead of beginning with a potential function and taking the gradient, we directly construct a vector field to use as the velocity command. We do this by smoothly combining locally-defined vector fields using bump functions, which guarantees that the resulting integral curves are smooth.

Stabilization of nonholonomic systems in obstacle-free environments has been studied in depth [2], [4], [7], and an approach for computing optimal trajectories is presented in [8]. When the environments are complex, however, the problem of global feedback control is difficult. Motion planning problems in robotics typically involve non-convex constraints resulting from obstacles in the environment, which presents a significant problem for traditional feedback control methods. One solution is to use state space sampling along with dynamic programming to achieve not only feedback, but approximately optimal trajectories [1], [14], [22]. This may be feasible for low-dimensional spaces, but both the time- and space-complexity is exponential in the dimension of the state space, assuming that the sampling resolution remains fixed.

As mentioned above, open-loop motion planning algorithms are excellent for finding feasible trajectories but do not produce global feedback plans. Some algorithms, such as RRTs [15] and PDST-EXPLORE [12], plan using system inputs and directly return a suitable trajectory for the system. Other methods such as PRMs [9] return kinematic paths which must be post-processed in order to yield trajectories for the system. While most sampling-based algorithms give only open-loop plans, this is not universally the case; one method that builds a feedback plan over the configuration space is the sampling-based neighborhood graph [24]. The SNG covers the free space with balls, each of which is equipped with a local navigation function which is guaranteed to convey the robot into a ball nearer to the goal state.

The most common approach to feedback motion planning in the presence of obstacles is based on potential fields. Khatib [10] developed a method which utilized a potential field over the operational space to guide a manipulator or mobile robot to the goal. This approach suffers from local minima, however, as do many potential field methods. Waydo and Murray give a stream function method for navigation in two-dimensional environments [23]. A highly influential potential field method is that of Rimon and Koditschek [19], who show how to develop *navigation functions* (potential functions with a unique minimum at the goal and meeting certain other criteria) using potential functions in a generalized sphere world.

Finally, Conner *et al.* [6] and Lindemann and LaValle [16] compute feedback plans over cell decompositions; the work of [16] forms the basis for this work. Conner *et al.* consider a cell complex environment in d-dimensional Euclidean space. They then impose a potential field over each individual cell, taking as the field the pullback of a potential function on a disk, which has a closed form solution. They require that the gradients of the potential fields be perpendicular to the cell boundaries, so that adjoining potential fields can be easily pieced together. Putting the individual "component control policies" together guarantees that the global control policy brings the robot to the goal. In addition to specifying a control policy for kinematic systems, they develop control policies for systems with dynamical constraints. Similarly, Lindemann and LaValle take a cell complex environment and define vector fields over the individual cells, which can

S. R. Lindemann is with the Department of Electrical Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA slindema@uiuc.edu

I. I. Hussein is Assistant Professor in the Department of Mechanical Engineering, Worcester Polytechnic Institute, Worcester, MA 01609 USA. ihussein@wpi.edu

S. M. LaValle is Associate Professor in the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801 USA. lavalle@uiuc.edu

be seen as component control policies. We describe this approach in detail below. Both [6] and [16] can be seen in the context of the sequential composition of funnels approach [17], in which a collection of controllers is developed, each of which converges to a goal set which is either the actual goal state or in the domain of another controller. Following a sequence of these controllers will cause the system to arrive at the goal state. This idea was further developed in [5], [20].

In the next section, we will describe and formulate our problem precisely. In Section III, we will describe our algorithm and prove that it solves the global navigation problem. Finally, we will discuss our algorithm and show examples in Section IV and conclude with Section V.

II. Problem Formulation

We consider the problem of a nonholonomic point robot navigating in a polygonal environment. We use a simple nonholonomic model given by

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v_f \cos\theta \\ v_f \sin\theta \\ v_\theta \end{pmatrix}. \tag{1}$$

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 general polygonal environment, $\mathcal{E}$, which is a bounded open subset of $\mathbb{R}^2$. By *general* we mean that the polygon may be nonconvex and may contain holes (i.e., be multiply-connected). The configuration space $\mathcal{C}$ is three-dimensional, consisting of position and orientation; formally, $\mathcal{C} = \mathcal{E} \times S^1$, in which $S^1 = [0, 2\pi]/\sim$, in which $\sim$ is an equivalence relation with $0 \sim 2\pi$. While we will use this three-dimensional model for the purposes of discussion, the controls are actually acceleration controls. In other words, the system can be viewed as a five-dimensional system satisfying the additional equations $\dot{v}_f = u_1$ and $\dot{v}_\theta = u_2$. The nonholonomic constraint can be expressed as $\dot{x}\sin\theta - \dot{y}\cos\theta = 0$.

The *goal point* is defined to be the point $x_g \in \mathcal{E}$. Our task is to construct a smooth feedback plan which solves the planar navigation problem, which will stabilize the system to the goal point while ignoring the orientation. Given a dynamical system, a *smooth feedback plan* is a vector field V on $\mathcal{C}$ which satisfies the following properties:

1) At every point, V is admissible (i.e., it satisfies the differential constraints of the system).
2) V is smooth except for a set of measure zero.
3) The integral curves of V are smooth.

By *smooth* we mean infinitely differentiable, i.e., C^∞. The vector field induces a mapping $\pi : \mathcal{C} \to \mathcal{U}$, in which $\mathcal{U}$ is the input space; since the vector field is admissible, this is guaranteed to be well-defined. A smooth feedback plan solves a navigation problem if all integral curves of the vector field converge to the goal region. In this work, the goal region is $x_g \times S^1$, for some $x_g \in \mathcal{E}$. In other words, we ignore the orientation of the robot and consider only its position. This is entirely reasonable; for the system we consider, Brockett's condition [3] 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.

III. Constructing Smooth Feedback Plans

In this section, we describe how to construct smooth feedback plans which satisfy the nonholonomic motion constraints while taking the robot to the goal region from any point in the environment. We begin by describing how to construct a smooth feedback plan over $\mathcal{E}$ as in [16]. We then use this in the construction of our smooth feedback plan over $\mathcal{C}$. The method of [16] is applicable in arbitrary dimensions; however, we will describe the two-dimensional case, since that is what we need for our problem.

A. Computing a Smooth Feedback Plan over $\mathcal{E}$

The feedback plan is computed as a plan for a point robot moving in a 2-dimensional cell complex in which each cell is a convex polygon. In our case, the complex results from the convex decomposition of the general polygonal environment $\mathcal{E}$. Denote the *goal state* by x_g, and the *goal cell* containing x_g by C_g. Using the connectivity of the convex cells, construct a graph and use a graph search algorithm (such as Dijkstra's algorithm) to determine a path from each cell to C_g. For each cell other than C_g there is a "successor" that represents the next cell on the path to the goal cell. Every cell with a successor is called an *intermediate* cell.

The global feedback plan is built by constructing vector fields over each cell, ensuring that they each have the desired properties, and guaranteeing that they match at the faces separating adjacent cells. The vector field of each cell must be smooth in the interior of the cell, and all integral curves must be smooth and exit the cell at the face shared with the cell's successor (except in the case of the goal cell; in this case, the integral curves must converge to the goal state).

Given a particular cell, define a vector field for each face of the cell (in the 2D case, each face is simply a one-dimensional edge). Denote these face vector fields by V_{f_i}. Face vector fields must have two properties:

1) A face vector field corresponding to an obstacle face must point away from the obstacle.
2) A face vector field corresponding to a face between adjacent cells must be consistent with the successor relation given above.

See Figure 1 for an illustration of an environment, the face vector fields, and a directed graph giving the cell path to the goal cell.

The vector field for the particular cell (denote this as V) must match the face vector fields $\{V_{f_i}\}$ at their respective faces and must smoothly interpolate between them in the interior of the cell. This is accomplished using an *attractor* vector field V_a, which is defined over the entire cell. The cell is partitioned into regions corresponding to each face; in the region corresponding to a particular face, the vector field V interpolates between the face vector field V_{f_i} and the

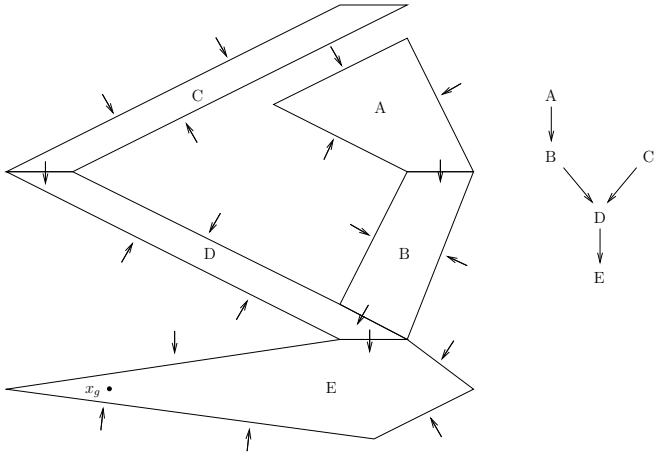


Fig. 1. An environment, corresponding face vector fields, and the graph showing how to reach the goal cell from any other cell.

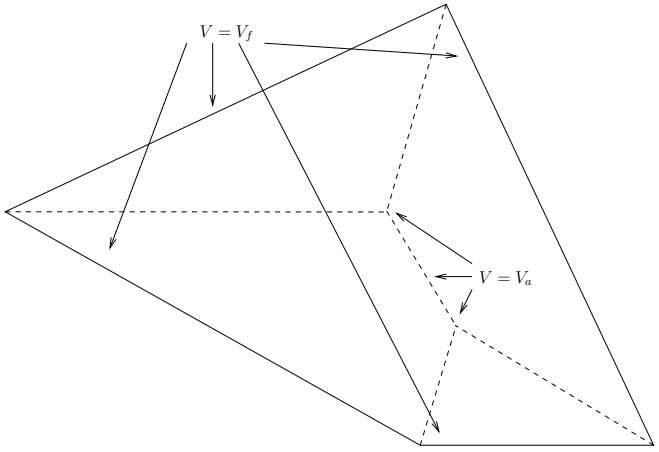


Fig. 2. An individual cell, partitioned using the generalized Voronoi diagram.

attractor vector field V_a. On the faces of the partition itself, $V = V_a$; as we have already said, on the face of the cell $V = V_{f_i}$. The partitioning method used is the generalized Voronoi diagram, or GVD. See Figure 2 for an illustration.

In order to smoothly interpolate between the face and attractor vector fields in the different partition regions, we use *bump functions*, which are defined as follows:

Definition 3.1: Let X be a smooth manifold, and let K be a closed set and U an open set, $K \subset U \subseteq X$. Then 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 $x \in K$.

For our purposes, we will use a bump function on the real line which transitions smoothly from 1 to 0 on the unit interval, which is defined as follows. Let $\lambda(s) = (1/s)e^{-1/s}$; then $b(s) = 1 - \lambda(s)/(\lambda(s) + \lambda(1 - s))$ on the unit interval, with $b(s) = 1$ for $s \leq 0$ and $b(s) = 0$ for $s \geq 1$. It is easily verified that this function is C^∞ over the real line. An illustration of such a bump function is given in Figure 3.

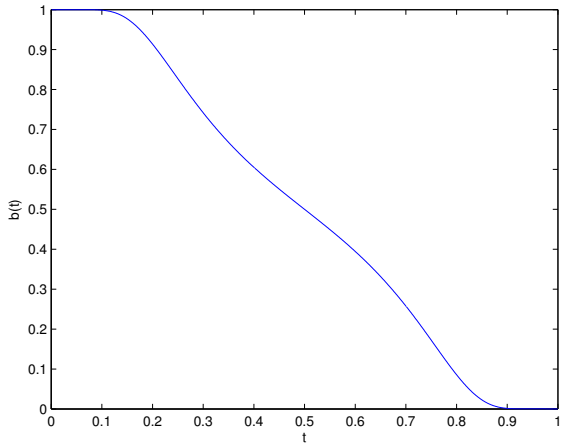


Fig. 3. A bump function.

The parameter used for the bump function is a product of analytic switches built using distances to the faces of the GVD. Formally, for any point p in the region corresponding to face f_x,

$$s(p) = 1 - \prod_{i=1}^{n} \frac{d(p, f_i)}{d(p, f_i) + d(p, f_x)}, \tag{2}$$

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

Since the bump function smoothly blends the face and attractor vector fields together, a vector field is obtained which is smooth over the interior of the cell. With small modifications, this approach can be used in the goal cell as well; piecing the cells together results in a smooth feedback plan. Using the shorthand $b(p) = b(s(p))$, we define the global vector field V at point p as $V(p) = \mathrm{norm}(b(p)V_f(p) + (1 - b(p))V_a(p))$, in which V_f is the face vector field for that point, V_a the attractor vector field, b the bump function, and norm is the normalization function, so that V is a unit vector field. Figure 4 shows an environment and a depiction of the computed smooth feedback plan. This method is extremely fast and can be computed in real time. While there is some extra overhead associated with dynamic environments, the algorithm is fast enough that it can be re-initialized in real time should the environment change.

B. A Smooth Feedback Plan Satisfying Nonholonomic Constraints

Using the vector field produced by the algorithm described above, we now construct a smooth feedback plan over the entire configuration space which will converge to the goal region while satisfying the constraints.

Recall that the free workspace is denoted $\mathcal{E}$, and the vector field V defines a smooth feedback plan over $\mathcal{E}$. In standard

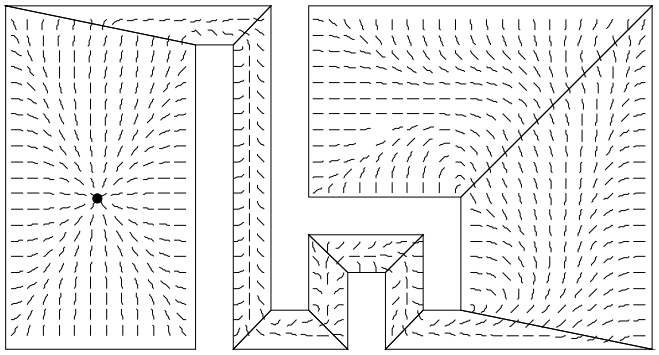


Fig. 4. An environment and a corresponding smooth feedback plan.

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* as $M_t = \{(p, \theta) \mid \theta = \theta_t(V(p)), p \in \mathcal{E}\}$. 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 $v_\theta = \frac{d}{dt}\theta(V(c(t)))$. It should be apparent that $\frac{d}{dt}\theta(V(c(t))) = \frac{d\theta}{dV}\frac{d}{dt}V(c(t)) = \frac{d\theta}{dV}\mathcal{L}_V V(c(t))$, in which $\mathcal{L}$ denotes the Lie derivative. 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. In contrast, our method 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). We accomplish this 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 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 $v_\theta = \frac{d\theta}{dV}\mathcal{L}_{\mathcal{P}(V)}V$. Note that on the target manifold, $V_n = V$. We define our orienting vector fields in the obvious way:

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

We now define the global vector field which avoids the obstacles, satisfies the nonholonomic constraints, and whose integral curves are smooth and converge to the goal region:

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

in which $b(s)$ is the bump function. We also need to define $s = \theta_{min}^{-1} \times \min\{|\theta - \theta_t|, |\theta - (\theta_t + \pi)|\}$. We define θ_{min} 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, $\theta_{min} \le \pi/2$. At this point, $V_n = 0$ because the original vector field is orthogonal to the orientation of the robot.

Second, the method of [16] 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 it is shown in [16] that V can be constructed so that $\epsilon = 1$, this means 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 only following the nominal vector field as much as necessary to reach the goal eventually. We favor the philosophy behind this approach, because it is decidedly not important to follow the original vector field; all that matters is that the robot reaches the goal. 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 configuration space.

C. Theoretical Properties

It remains to verify that the vector field W possesses all the properties we desire. We do this in the following theorems.

Theorem 3.2: 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 non-smooth portions of W are a set of measure zero, and all integral curves flow away from the non-smooth portions and are consequently smooth. ■

Theorem 3.3: 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 3.4: The integral curves of W never lead to obstacle collision.

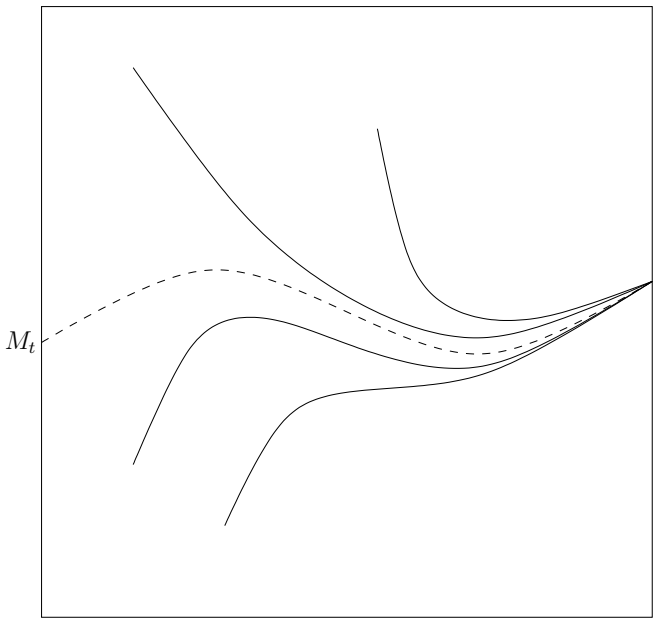


Fig. 5. An illustration of convergence to the target manifold, M_t.

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 state. 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 3.5: 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: Consider only the θ coordinate, and use a Lyapunov function $L(p) = 1/2(\theta - \theta_t)^2$. This has a unique minimum on the target manifold, where $(\theta - \theta_t) = 0$. From the construction of W, one can derive the result $\dot{L} = -(1 - b(s))|\theta - \theta_t|$, which is everywhere positive except on the target manifold. Hence θ_t is globally asymptotically stable (the integral curves converge to the target manifold). To be precise, it will converge to one of the target manifolds (recall that there was one target manifold corresponding to moving forward and another corresponding to moving backward). Figure 5 illustrates this. ■

Lemma 3.6: 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 $x(c_1(t_0)) = x(c_2(t_0)$

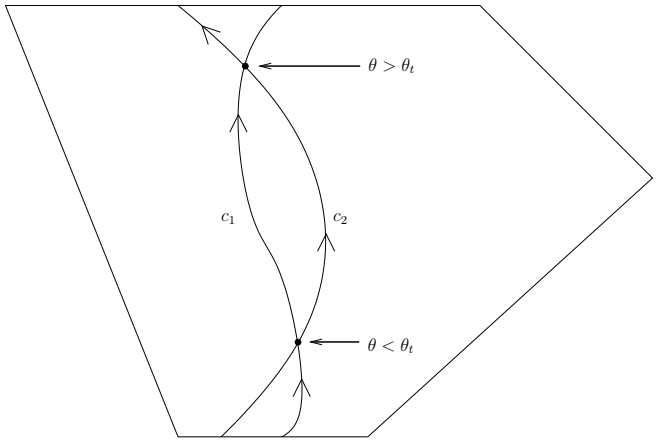


Fig. 6. An impossible situation. The integral curve c_1 partitions the cell into two halves; crossing from one side to the other at a given point implies a relationship between the relative orientations at that point.

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 6 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 3.7: 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 [16], we show 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 W cannot cross any flow from V more than once (Lemma 3.6), 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 gets 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 3.5 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 [16], the integral curve cannot be trapped and must exit the cell by the exit face.

We have shown that all integral curves of W must reach the goal cell. A similar argument shows that 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 $\dot{V}(x_g - p) > 0$ 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 non-goal 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. ■

IV. Analysis and Examples

From a practical standpoint, our 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. Our extension to nonholonomic systems in this work has the same algorithmic complexity and practical time requirements as the previous work in [16]. First, the component vector fields must be computed for the given polygon. If the environment is given as a general polygon, then it must first be decomposed into convex pieces. In two dimensions, this is algorithmically straightforward and extremely fast. Using a modified version of Seidel's algorithm [18], [21], this can be done in expected $O(n \log^* n)$ time, in which n is the geometric complexity of the polygon (i.e., the number of vertices). After decomposing the polygon into convex pieces, the cell connectivity graph must be explored to generate the successor of each cell. Using simple breadth first search, this can be done in $O(n)$ time. Additionally, the point location data structure should be built, which requires $O(n)$ time [11]. The face and attractor vector fields can also be computed in linear time. Second, when a new initial state is given to the algorithm, then a point location query must be performed. This can be done in $O(\log n)$ time, since the preprocessing has already been performed. Finally, after the initialization is complete, only linear time is required to compute the vector field value (in the complexity of the cell along with one neighboring cell).

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 re-performed; 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, we have included several examples of the system trajectories of our algorithm. As we have said earlier, there is a fair amount design freedom in choosing how aggressively to approach the target manifold. Figure 7 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 8, which indicates the orientation of the robot at different points along its trajectory. Figure 9 shows a similar trajectory; the orientation annotations indicate the orientation of the nominal (not actual) vector field at those points. The accompanying plot shows the convergence to the target manifold, i.e., $\theta \to \theta_t$.

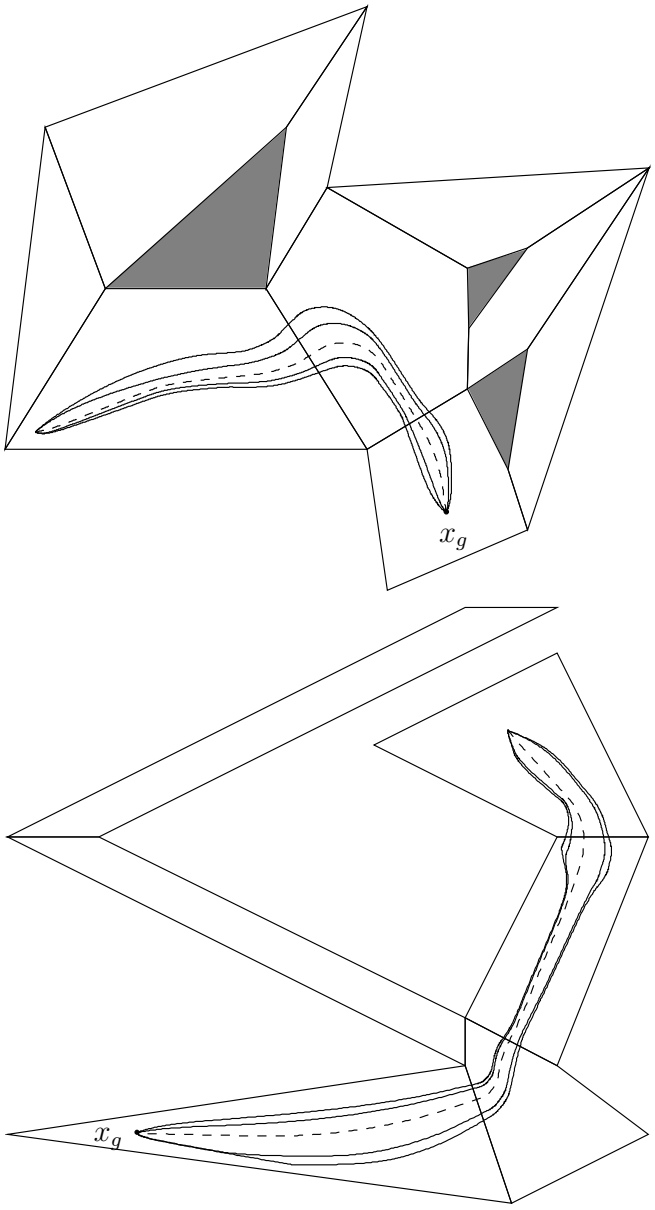


Fig. 7. Two environments, with goal states x_g and trajectories from an initial point with initial angular deviation.

V. Conclusion

In conclusion, we have presented an algorithm for constructing a smooth feedback plan for a nonholonomic point robot in a polygonal environment. The integral curves of the corresponding vector field are smooth, satisfy the nonholonomic constraints, and stabilize the robot to the goal region $x_g \times S^1$. We construct the feedback plan using a nominal plan computed as in [16], which defines a target manifold in the configuration space. We guarantee that the the integral curves converge to the target manifold by smoothly interpolating between trying to follow the nominal vector field (after projecting them onto the nonholonomic constraint distribution) and reorienting the robot to improve alignment with it. By guaranteeing that the integral curves converge to the target manifold, we prove that the integral curves converge to the goal region as well. Our method is extremely fast to compute and is suitable for real-time application.

In the future, we intend to address systems with more complex nonholonomic constraints. While the differential drive model is extremely common and thereby important, there are many other systems of interest which we would like to develop smooth feedback plans for, such as car-like robots with path curvature constraints. We would also like to create feedback plans for systems modeled as polygonal robots in addition to point robot models.

Acknowledgments

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

References

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

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

[3] R. W. Brockett. Asymptotic stability and feedback stabilization. In R. W. Brockett, R. S. Millman, and H. J. Sussmann, editors, *Differential Geometric Control Theory*. Birkhäuser, Boston, MA, 1983.

[4] F. Bullo and A. Lewis. *Geometric Control of Mechanical Systems*. Springer-Verlag, New York, NY, 2004.

[5] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *Int. J. Robot. Res.*, 18(6):534–555, 1999.

[6] D. C. Conner, A. A. Rizzi, and H. Choset. Composition of local potential functions for global robot control and navigation. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 3546–3551, 2003.

[7] J. Cortés, S. Martínez, J. P. Ostrowski, and H. Zhang. Simple mechanical control systems with constraints and symmetry. *SIAM Journal on Control and Optimization*, 41(3):851–874, 2002.

[8] I. I. Hussein and A. M. Bloch. Optimal control of underactuated nonholonomic mechanical systems. In *Proceedings of the 2006 American Control Conference*, 2006. To appear, preprint available at http://arxiv.org/abs/math/0510009 .

[9] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. & Autom.*, 12(4):566–580, June 1996.

[10] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Robot. Res.*, 5(1):90–98, 1986.

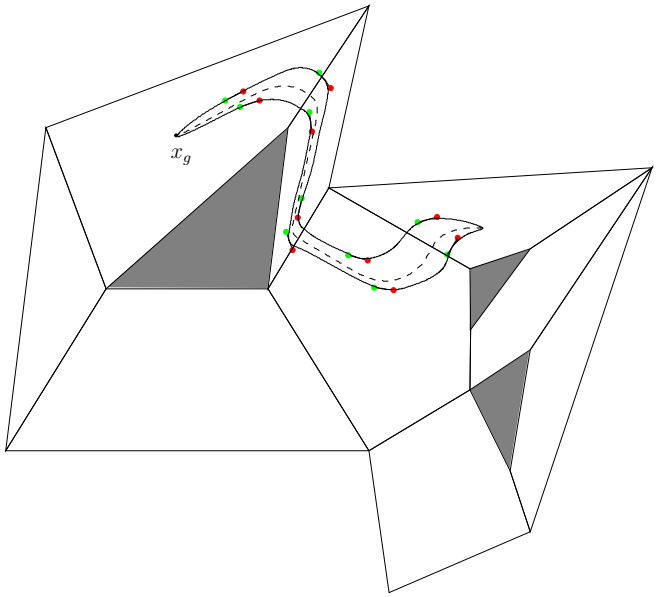


Fig. 8. Two trajectories from an initial point, with indications of robot orientation.

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

[12] A. M. Ladd and L. E. Kavraki. Fast exploration for robots with dynamics. In *Proc. Workshop on Algorithmic Foundation of Robotics*, pages 313–328, 2004.

[13] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. (Also available at http://msl.cs.uiuc.edu/planning/).

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

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

[16] S. R. Lindemann and S. M. LaValle. Smoothly blending vector fields for global robot navigation. In *Proc. IEEE Conference on Decision and Control*, pages 3553–3559, 2005.

[17] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor. Automatic systhesis of fine-motion strategies for robots. *Int. J. Robot. Res.*, 3(1):3–24, 1984.

[18] A. Narkhede and D. Manocha. Fast polygon triangulation based on seidel's algorithm. In A. Paeth, editor, *Graphics Gems V*. Academic Press, Boston, 1995.

[19] E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential fields. *IEEE Trans. Robot. & Autom.*, 8(5):501–518, October 1992.

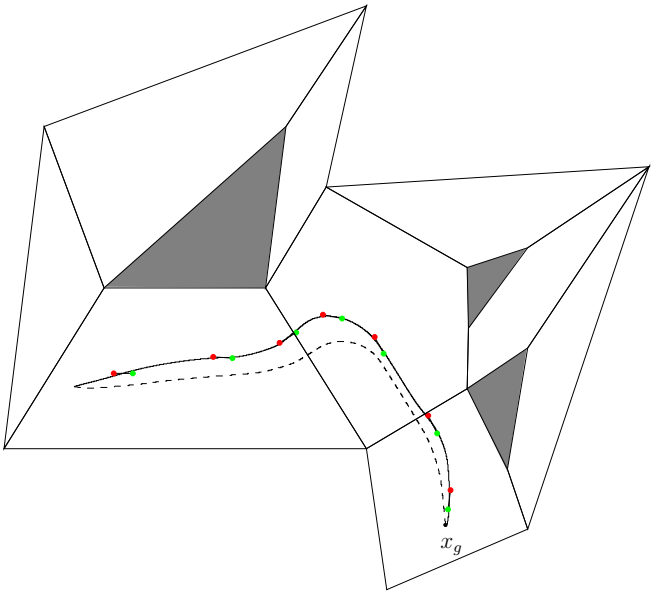
[20] A. A. Rizzi. Hybrid control as a method for robot motion programming. In *IEEE Int. Conf. Robot. & Autom.*, pages 832–837, 1998.

[21] R. Seidel. Small-dimensional linear programming and convex hulls made easy. *Discrete Comp. Geom.*, 6:423–434, 1991.

[22] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Trans. Autom. Control*, 40(9):1528–1538, September 1995.

[23] S. Waydo and R. M Murray. Vehicle motion planning using stream functions. In *IEEE Int. Conf. Robot. & Autom.*, pages 2484–2491, 2003.

[24] 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*, 2003.



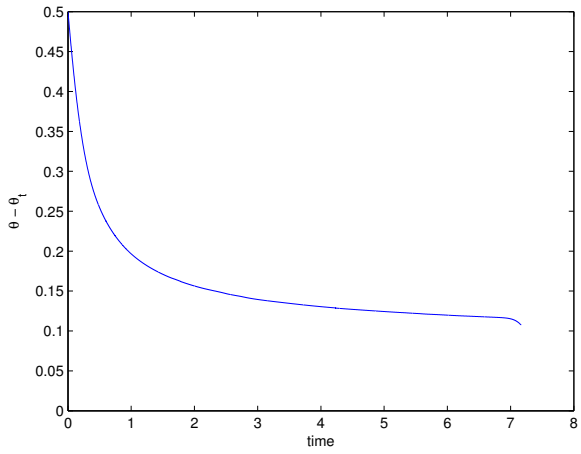


Fig. 9. An illustration of convergence to the nominal vector field. The annotations indicate the difference between the actual vector field and the nominal one. The angular error decreases monotonically; the trajectory ends when the robot reaches a small neighborhood of the goal region.