Continuous Planning With Winding Constraints
Using Optimal Heuristic-Driven Front Propagation

Dmitry S. Yershov†  Paul Vernaza‡  Steven M. LaValle†

Abstract—Recent work has produced methods to solve the winding-constrained optimal feedback navigation problem. Given the start and the goal positions and the winding constraints, the solution to this problem is a feedback vector field such that, when integrated from the start, the trajectory is the shortest path connecting the start and the goal which satisfies given constraints. Such constraints intuitively restrict the direction and the number of times the path winds around given planar regions. We formulate a continuous version of this problem that contrasts with the discrete treatments previously presented. This leads to a geometrical characterization of the problem for which simplicial complex approximation is particularly useful. Thus, it yields theoretical insight as well as a practical algorithm for approximating the continuous problem using an efficient and high-accuracy heuristic-driven front propagation method on simplicial meshes. Experimental results are given evaluating the solution quality and efficiency of the method versus methods based on the discrete formulation and without using heuristics.

I. INTRODUCTION

Recent work has posed the following topologically-constrained navigation problem: given two points in the plane, find the shortest path connecting them that satisfies winding constraints [3], [4], [18]. These winding constraints specify in what way the path should wind around specified regions of the plane. An example of a problem that can be formulated in such a way is planning for surveillance, as illustrated in Fig. 1. Here, winding constraints can be enforced to ensure that the generated path circles around specified regions of interest in a particular way.

Previous approaches [3], [8], [18] have formulated this problem in terms of combinatorial search: a discrete graph is built to represent the connectivity between position and winding states, and an optimal path is found in this graph via a heuristic-driven search algorithm such as A*. The aim of this work is to formulate and solve the winding-constrained planning problem in a more natural continuous setting. Such an approach has several advantages. First, it yields important insight into the underlying geometry of the problem. In fact, we are able to visualize the configuration space in a way that makes obvious some of the pertinent representational and computational issues that arise in solving the problem, as seen in Figures 3 and 4. Secondly, this insight leads to a practical algorithm for solving the problem. Once the geometry of the configuration space is known, it can be constructed explicitly in a way that allows solution via high-accuracy front-propagation methods. In contrast to discrete formulations, this approach has the advantage of generating paths that are free of discretization artifacts that degrade solution quality. Finally, we will show that it is possible to obtain these advantages without significant loss in computational efficiency; in particular, we can take advantage of a recently developed A*-like front propagation method that retains optimality of solutions [19].

II. RELATED WORK

Our work can be considered a refinement of various recent methods involving planning trajectories subject to topological constraints. Most relevant are those methods that employ graph-based search algorithms. Bhattacharya et. al. [3] introduced one such method in which graph-based search for point-to-point navigation was augmented with a complex-valued state encoding information about path winding. Simplified variants using more direct encodings of winding were presented in [8] and [18] and applied to vision-based tracking and planning loops for robotics applications, respectively. Unfortunately, these discrete, graph-based constructions obscure the intrinsic geometry of the problem. The continuous formulation presented here reveals the geometry in a way that aids intuition and enables the use of efficient and highly accurate numerical methods.

We identify the configuration space of the winding-constrained planar navigation problem as a covering space of the multiply punctured plane. A similar construction

† D. S. Yershov and S. M. LaValle are with the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801.
‡ P. Vernaza is with The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 19104
appeared in [12], in which a planning algorithm for a robot attached to a cable was developed. However, that work employs a graph-based discretization of the configuration space that additionally abstracts away details of specific winding angles; we consider a continuous version of this structure that encodes winding angles, allowing us to plan with explicit winding constraints. Also related is work from computational geometry on the problem of finding the shortest path homotopic to a given path (or set of paths) such as [2], [6], [11]. Our work differs from these in that we seek paths subject to winding constraints, which are equivalent to homology constraints for our domain (as opposed to homotopy constraints) [10]. We also employ numerical methods that allow us to easily use spatially varying, non-Euclidean metrics that often arise in path planning for robotics.

III. WINDING-CONSTRAINED OPTIMAL FEEDBACK NAVIGATION PROBLEM

The basic idea of our work is to reduce the problem of winding-constrained planning in the plane to that of finding a minimum-cost path on a surface embedded in \( \mathbb{R}^3 \). Once this is accomplished, it is straightforward to apply any of several numerical methods to find such a path; however, the underlying combinatorial nature of the problem provides a challenge for classical Dijkstra-like methods. In the next section, we then show how the underlying two-dimensional geometry of the problem allows us to apply recently developed heuristic-driven methods that are significantly more efficient.

A. Problem formulation

In intuitive terms, our problem may be expressed simply as that of finding the minimum-cost continuous path in the plane that satisfies winding constraints while also avoiding obstacles. We assume the winding constraints are defined with respect to points located within obstacle regions of nonzero area. Fig. 2b shows some examples of winding-constrained paths.

A more formal definition of the problem is as follows. We regard a path as a continuous function \( I \rightarrow \mathbb{R}^2 \setminus \mathcal{O} \), where \( I \) represents the unit interval and \( \mathcal{O} \) is an open subset of \( \mathbb{R}^2 \) representing obstacle locations (or holes) in the environment. We assume that some holes are associated with constraints describing how the path should wind around them. These constraints are expressed in terms of winding angles [14], the concept of which is illustrated in Fig. 2a. Winding constraints are encoded as vectors in \( \mathbb{R}^N \) that specify desired winding angles to achieve for each of \( N \) predetermined holes. Start and end winding angle vectors denoted by \( s_\theta \in \mathbb{R}^N \) and \( g_\theta \in \mathbb{R}^N \). Start and end locations are denoted by \( s_x \) and \( g_x \) respectively. Our objective is specified in terms of a cost function \( C : \mathbb{R}^2 \rightarrow \mathbb{R}^+ \) (where \( \mathbb{R}^+ \) represents the positive reals). We then wish to find \( \hat{x}^* \), defined as the solution to the following optimization problem:

\[
\hat{x}^* = \arg \min_{\hat{x} : I \rightarrow \mathbb{R}^2 \setminus \mathcal{O}} \int_0^1 C(\hat{x}(t)) \|\dot{\hat{x}}(t)\| dt \quad (1)
\]

subject to

\[
\hat{x}(0) = s_x, \quad \hat{x}(1) = g_x
\]

\[
\hat{\theta}(0) = s_\theta, \quad \hat{\theta}(1) = g_\theta
\]

Implicit in this optimization problem is another constraint coupling the optimized path \( \hat{x} \) in space to the dependent path of winding angles \( \hat{\theta} : I \rightarrow \mathbb{R}^N \); the latter is dependent on the former in the sense that the winding angles are completely determined by the spatial path.

B. The geometry of winding-constrained planning

We first explore the geometry of the configuration space in the case that only one winding constraint is present. Fig. 3 shows a visualization of the configuration space for this problem. Since only one winding constraint is present, it suffices to keep track of one winding angle for the purpose of planning. The winding angle is visualized along the vertical direction in the figure.

In the topological sense, this kind of construction is known as a covering space [10]. A covering space is everywhere locally similar to the base space, which is in this case a plane with a single hole (representing an obstacle) removed. The covering space in this case is constructed as a surface with the property that points on the surface correspond to valid combinations of position and winding states. Furthermore, paths on the surface correspond to feasible paths through the joint space of position and winding. Therefore, we can find an optimal path subject to a single winding constraint.
by finding an optimal path lying on a surface such as that illustrated in Fig. 3.

The general case appears significantly more complex, as the configuration space generally contains winding information about multiple holes. However, we will see that it is again possible to represent the configuration space in this case as a surface embedded in \( \mathbb{R}^2 \). The construction relies on borrowing a trick from [18] in which all of the winding angles \( \theta_i \) are reversibly encoded into a single real value \( \Theta \). This idea is illustrated in Fig. 4b, in which we have visualized the configuration space generated by our method of a planning problem with two winding constraints. Here we see that every possible combination of winding angles at a given location \( x \) is represented by a unique point on a covering space of the twice-punctured plane. It is perhaps counterintuitive that an object can exist that simultaneously encodes such a combinatorial property while retaining the character of a continuous surface; however, this is indeed a surface even in the general case, as will be shown shortly. It is instructive to attempt to trace out different paths on the surface to understand how the surface manages to encode different winding states without intersecting itself anywhere.

The key to constructing surfaces of this type is to use the encoding method of [18] in which \( \Theta \) is simply a linear combination of winding angles, where the coefficients are chosen to be rationally independent. For example, we may choose the coefficients to logarithms of distinct prime numbers \( z_i \):

\[
\Theta = \sum_i \theta_i \log z_i. \tag{2}
\]

This encoding ensures that the winding angles \( \theta_i \) can be uniquely recovered from the \( \Theta \)-value; this is possible because the possible winding angles at each location in the plane constitute a countably infinite set [18].

We can then characterize feasible trajectories in \((x, \Theta)\) space as those that obey the dynamics of the following underactuated system, where \( \tilde{u} : [0, t_f] \rightarrow U \), in which \( U = \{u \in \mathbb{R}^2 \mid \|u\| = 1\} \), is a control input trajectory:

**Definition 3.1 (Joint position and winding dynamics):**

\[
\begin{align*}
\dot{x}(t) &= \tilde{u}(t) \tag{3} \\
\dot{\Theta}(t) &= \sum_{i=1}^N \theta_i(\dot{x}(t), \tilde{u}(t)) \log z_i \tag{4}
\end{align*}
\]

This definition allows us to draw a formal correspondence between the configuration space in \((x, \Theta)\) coordinates and a surface embedded in \( \mathbb{R}^3 \).

**Theorem 3.1:** Assume \( |\tilde{u}| < M, \forall i \) for some \( M \in \mathbb{R} \). Then the set of all feasible trajectories of the dynamical system in Def. 3.1 constitutes a two-dimensional manifold.

**Proof:** (Sketch) The interesting part of the proof is to construct bijections (i.e., charts) between patches of feasible configurations and \( \mathbb{R}^2 \). If the patches are chosen to be sufficiently small, the projection \((x, \Theta) \mapsto x\) constitutes a suitable bijection. The proof of this relies on the fact that there exists a minimum separation in \( \Theta \)-value between feasible configurations with the same location in \( x \), which arises due to the bounds on \( |\theta_i| \), the nonzero areas of the regions about which the path is wound, and the rational independence of the winding coefficients.

The intuition for the proof is as follows. Imagine we are walking through a parking garage. The \( \Theta \)-value represents our elevation, and \( x \) represents our horizontal coordinates. For each starting location, we consider a patch of the parking garage consisting of all \((x, \Theta)\) states reachable from the start in \( \tau \) seconds or less. Our task is to show that for small enough
the elevation (Θ) of any location reachable in time $t < \tau$ is uniquely determined by its position $x$; that is, we can never reach the same $x$ at another level in the garage as long as we walk a sufficiently small distance. We can conclude this is true based on two facts: first, that our distance traveled in $Θ$ is bounded by a function of $τ$; and second, that there exists a minimum elevation difference between levels of the parking garage. Therefore, if we choose $τ$ such that we are sure that our deviation in $Θ$ is less than the minimum elevation difference between levels of the garage, we can state with certainty that we have not arrived at the same $x$ via a different level of the garage. Hence, the projection map $(x, Θ) \rightarrow x$ is bijective when restricted to the patch.

C. Hamilton-Jacobi-Bellman equation

The optimization problem (2) is equivalent to the optimal control problem for the underactuated system (Definition 3.1). In this setting, the cost functional is defined as

$$L(\bar{x}, \bar{Θ}, \bar{u}) = \int_{0}^{t_f} C(x(t))\|\bar{u}(t)\| \, dt, \quad (5)$$

if $\bar{x}(t_f) = g_x$ and $\bar{Θ}(t_f) = g_{Θ}$, and is infinite, otherwise. Note that argument of the cost functional $Θ$ is not explicitly present inside the integral. Nevertheless, the dependence on $Θ$ is carried implicitly in the structure of the manifold on which the system is constrained to lie.

Let the initial position vary on the configuration space; that is, $s_x = x$ and $s_{Θ} = Θ$ for some feasible $x$ and $Θ$. We consider the optimal cost-to-go function $V(x, Θ) = \min_{\bar{u}} L(\bar{x}, \bar{Θ}, \bar{u})$, which represents the minimal trajectory cost starting at $x$ and $Θ$ and arriving to $g_x$ and $g_{Θ}$. Using Bellman’s dynamic programming principle [1], which can be restated as “the tail of the optimal trajectory is itself optimal”, we derive the Hamilton-Jacobi-Bellman (HJB) equation for the cost-to-go function:

$$V(x, Θ) = \lim_{\Delta t \rightarrow 0} \inf_{u \in U} \left\{ \int_{0}^{\Delta t} C(x + ut) \, dt + V(x + \Delta x, Θ + \Delta Θ) \right\}, \quad (6)$$

in which $Δx = uΔt$ and $ΔΘ = \sum \Delta θ_i \log z_i$. In the limit of $Δt \rightarrow 0$, the equation above is equivalent to a partial differential equation in the local coordinates of the chart that contains point $(x, Θ)$:

$$0 = \min_{u \in U} \left\{ C(x)\|u\| + \nabla_x V(x, Θ) \cdot u \right\}. \quad (7)$$

Once the optimal cost-to-go function is computed, a feedback policy function, $π : S \rightarrow U$ is given as a minimizer of the right-hand-side of (7):

$$π(x, Θ) = \arg \min_{u \in U} \left\{ C(x)\|u\| + \nabla_x V(x, Θ) \cdot u \right\}. \quad (8)$$

In this sense, the cost-to-go function can be considered an optimal version of a navigation function [15]. Note that in the local coordinates parameters of the equations above are independent of $Θ$, although $Θ$ determines which chart of the configuration space is used to define a local projection. Thus, the cost-to-go function, $V$, and the optimal feedback policy, $π$, depend on $Θ$, and they can take different values at the same point $x$ of the environment.

Finally, to find a solution of (7), we impose boundary conditions at $g_x$ and $g_{Θ}$. Clearly, once the robot is at the goal with correct winding angles the cost-to-go is zero. Thus, $V(g_x, g_{Θ}) = 0$ is used as the boundary condition. Moreover, to guarantee a collision-free trajectory we require $V(x, Θ) = \infty$ for all $Θ$ if $x \in X_{obs}$.

Equation (7) usually does not have a closed-form solution. Therefore, we resort to numerical methods in order to find an approximation of the cost-to-go function. This leads to

Fig. 4. Visualization of path planning with winding constraints for a simple environment with two holes (refer to caption of Fig. 3 for interpretation). Path is constrained to wind once counterclockwise around the left hole and once clockwise around the right hole. Vertical coordinate represents $Θ$-value, as defined by Eq. (2).
an approximate feedback policy, which, when integrated, produces an approximation of the shortest path that satisfy winding angle constraints.

IV. NUMERICAL ALGORITHMS

A. Simplicial Methods

Simplicial complexes are widely used in computational physics and numerical methods to approximate smooth differential manifolds. Thus, motivated by Theorem 3.1, we discretize the configuration space using a two-dimensional simplicial complex, which is defined as a tuple $(S_d, \mathcal{T})$. Here, $S_d = \{(x,\Theta)\}_{i=1}^N$ is a finite set of vertices sampled from the configuration space, and $\mathcal{T}$ is an abstract two-dimensional complex; see definitions in [19] and references therein. Intuitively, two-dimensional simplicial complex is a triangular tessellation of a two-dimensional manifold.

Let $\hat{C}$ be a piecewise constant discretization of $C$ on a simplicial complex, that is $\hat{C}$ takes a constant value on simplex $S_T$ equal to that of $C$ in the simplex center and denoted as $C_T$. Also let $\hat{V}$ be a piecewise linear discretization of $V$, such that $\hat{V}(x,\Theta) = \sum_{i \in \mathcal{T}} \alpha_i V(x,\Theta_i)$ for $(x,\Theta) = \sum_{i \in \mathcal{T}} \alpha_i (x,\Theta_i)$. Function $V$ is uniquely defined by its values at vertices of the simplicial complex, which we denote as $\hat{V}_i$ for simplicity. Considering (6) in the neighborhood of $(x,\Theta_i)$, we derive a discrete HJB equation

$$\hat{V}_i = \min_{T \in \text{St}(i)} \inf_{x \in S_T} \left\{ \hat{C}_T \|x_i - \sum_j \alpha_j x_j\| + \sum_j \alpha_j \hat{V}_j \right\}, \quad (9)$$

in which $\text{St}(i) = \{T \in \mathcal{T} \mid i \in T\}$ is a star of vertex $i$. The discrete boundary conditions are imposed in simplex $T$, for which $(g_x, g_\Theta) \in S_T$, and are such that $\hat{V}(g_x, g_\Theta) = 0$. Finally, the discretized feedback policy is derived from (8) and (9)

$$\hat{\pi}(x,\Theta) = \arg \min_{u \in U} \{ \hat{C}(x)\|u\| + \nabla_x \hat{V}(x,\Theta) \cdot u \}. \quad (10)$$

Note that $\hat{C}$ and $\nabla_x \hat{V}$ are piecewise constant functions and so is the discrete feedback policy.

Equation (9) defines a system of nonlinear equations with respect to $\{\hat{V}_i\}_{i=1}^N$. This system can be solved efficiently in one “sweep” through the simplicial complex using the Simplicial Dijkstra Algorithm (SDA) [19]. The SDA is a generalization of the Dijkstra’s graph search algorithm [5] to arbitrary simplicial complexes. It also generalizes Fast Marching methods for front propagation problems in Physics [13], [16], and interpolation-based methods that use grid discretization for the shortest path problem [7], [17].

B. Heuristic-Driven Algorithms

Dijkstra’s graph search algorithm is an omnidirectional front propagation algorithm. Hence, it is unaware of extra information, in case the initial robot position is known. On the other hand, in [9] Hart and others introduced the A* algorithm, which employs a heuristic to focus costly computations in the derivation of the initial position. Moreover, if the heuristic is admissible and consistent, A* algorithm is guaranteed to compute the shortest path, exploring provably minimal number of vertices. In [19] the authors presented the Simplicial A* Algorithm, which under similar conditions, computes the optimal cost-to-go function on a simplicial complex. In the algorithm outlined below, $H_i$ represents the value of the heuristic at vertex $i$.

**Input**: Simplicial complex, $(S_d, \mathcal{T})$, initial and goal position, $s_x$ and $g_x$, and initial and goal winding angles, $s_\Theta$ and $g_\Theta$.

**Output**: Approximations $\hat{V}$ and $\hat{\pi}$

1. Initialize set $Q$ of “open” as $T$, such that $(g_x, g_\Theta) \in S_T$
2. Initialize set of labels $\{\hat{V}_i\}_{i=1}^N$, such that $\hat{V}_i \leftarrow \|x_i - g_x\|$ for $i \in Q$, and $\hat{V}_i \leftarrow \infty$ otherwise
3. while $Q$ is not empty do
   4. Pop $j$ from $Q$ with the lowest value of $\hat{V}_j + H_j$
   5. for all $T \in \text{St}(j)$ do
      6. for all $i \in T \setminus \{j\}$ do
         7. $(V^*, \pi^*) \leftarrow \text{update}(i, T)$
         8. if $V^* < \hat{V}_i$ then
            9. $\hat{V}_i \leftarrow V^*$; $\pi_T \leftarrow \pi^*$
      10. Push $i$ into $Q$ if $i \notin Q$

Careful choice of the heuristic is important to improve the performance of the planning algorithm and guarantee the optimality of the feedback policy. In previous work [18], the maximum of minimum-length excursions, which satisfy at least one constraint, was successfully implemented to solve the considered problem using the A* graph search algorithm. On the other hand, in [19] the rescaled Euclidean distance function was introduced to satisfy admissibility and consistency of the heuristic for the Simplicial A* algorithm. In this paper, we use a combination of these two ideas: the rescaled maximum of minimum-length excursions is implemented to construct an admissible and consistent heuristic for the SAA.

V. RESULTS AND DISCUSSION

The proposed simplicial decomposition algorithms for feedback planning with winding angle constraints were implemented and evaluated in numerical experiments. As discussed in Section III, Figures 3 and 4 show the result of applying our algorithms to two simple winding-constrained planning problems. Note that the figures show the actual simplicial decompositions constructed by the algorithms. Fig. 3 was generated using SDA, while Fig. 4 was generated using SAA.

We also implemented the graph-based method of [18] in order to compare the quality of the solutions generated thus to those generated by our method. The graph-based method constructs and searches a graph on nodes consisting of valid joint position and winding states $(x,\Theta)$; i.e., from a starting state $(x,\Theta)$, we recursively generate successor states of the form $(x + \Delta, \Theta(x + \Delta))$, where $x$ and $x + \Delta$ are adjacent points on a regular grid in position space. In our experiments, each grid point was considered adjacent to its eight nearest neighbors. We compared the paths generated in this way to the paths generated by our simplicial-decomposition-based methods for the previously discussed examples. The results are shown in Fig. 5. Since we used a uniform cost function in
this experiment, the optimal solution consists of portions of obstacle boundaries and straight line segments. The paths generated by the simplicial methods clearly exhibit this property with minimal discretization artifacts. By comparison, the graph-based method generated solutions with large discretization artifacts, as expected.

We also studied the effect of the heuristic on running time. Fig. 6a shows the surfaces generated by the SDA (left) and the SAA (right) algorithms under identical initial conditions. As we see from the figure, the SAA explored only a fraction of the configuration space compared to that of the SDA. It was mentioned earlier that the SDA is an omnidirectional method. Moreover, the information as to whether \( \Theta \) must increase or decrease in order to arrive at the goal is not encoded in the cost functional, and is thus unavailable to the algorithm. Therefore, the SDA must in theory compute the cost-to-go function for at least twice as many nodes as the SAA. In the experiment, however, the SDA explored close to 60 thousand nodes, whereas the SAA explored slightly over 10 thousand nodes.

An experiment applying our method to UAV surveillance was also performed, borrowing a scenario found in [18]. Here, winding constraints are used as a surrogate for the constraint that the UAV should obtain 360-degree views of certain regions of interest. The result shown in Fig. 1 demonstrates that the method is applicable to problems of realistic complexity with non-uniform cost functions.

Finally, the scalability of the algorithms was analyzed in a simple environment by plotting running time as a function of the number of winding constraints introduced. The graph in Fig. 7b shows the running time of both the SAA and SDA algorithms. Both algorithms scale exponentially as the number of ROI increases. However, the SAA algorithm is faster by a factor of four, approximately, for up to five winding constraints. After the fifth winding constraint, SDA slowed considerably; hence, those results are not included in the graph. Therefore, as expected, the heuristic-driven nature of SAA is of considerable benefit in solving this type of problem.

VI. Conclusion

In conclusion, we have developed a continuous version of the problem of optimal path planning in the plane subject to winding constraints, described the geometry that arises hence, and shown how recently developed heuristic-driven front propagation methods may be applied to obtain highly accurate solutions. We formulated the problem as an optimal control problem for an underactuated system, which gave insight into the configuration space of the original system. We proved that, regardless of the number of regions of interest, the configurations space is a two-dimensional topological manifold, which can be embedded in \( \mathbb{R}^3 \) without self-intersections. Additionally, we constructed numerically and illustrated two configuration spaces with one and two regions of interest. Further, using a simplicial complex to discretize the configuration space, we proposed a numerical method that computes an approximation of the optimal feedback policy for the underactuated system. We have shown that the shortest path can be computed efficiently using the heuristic-driven Simplicial A* algorithm instead of standard fast marching methods for front propagation. Finally, the presented algorithm computes the approximate shortest path
with considerably fewer discretization artifacts, compared to methods based on graph search.

ACKNOWLEDGMENT

The authors were partially supported by ONR DR-IRIS MURI grant #N00014-09-1-1052. LaValle is supported in part by NSF grants 0904501 (IIS Robotics) and 1035345 (Cyberphysical Systems).

REFERENCES


