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.

Whereas previous approaches [3], [8], [18] have solved this problem via graph search, we 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 present an equivalent three-dimensional underactuated system that is "trapped" on the configuration space of the original problem. Second, this insight leads to a useful algorithm, since the configuration space can be constructed explicitly, allowing us to apply front-propagation methods that lack the discretization artifacts inherent in graph-based approaches. Finally, we show that it is possible to obtain these advantages without significant loss in computational efficiency; in particular, we take advantage of a recently developed A*-like front propagation method that retains optimality of solutions [19].

[†] 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

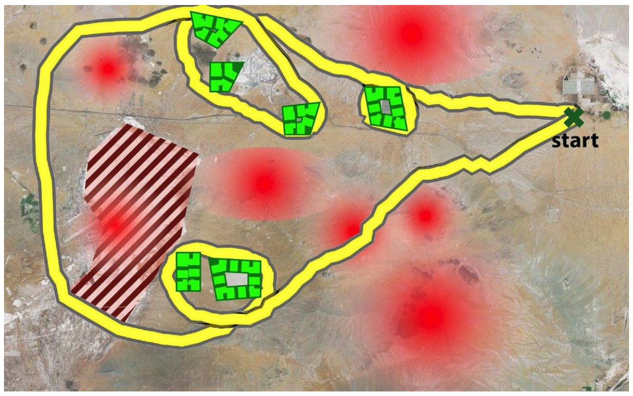
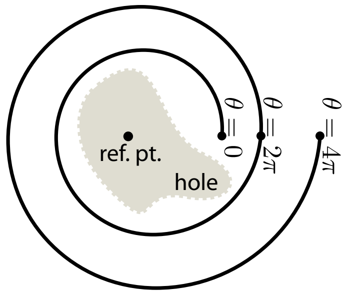


Fig. 1. Result of applying our method to the problem of winding-constrained planning for UAV surveillance. UAV path depicted as yellow line. Red areas constitute radar installations of high traversal cost (with increasing cost towards ellipse centers). Green dashed lines denote areas subject to winding constraints—UAV is constrained to wind around these regions in order to observe them.

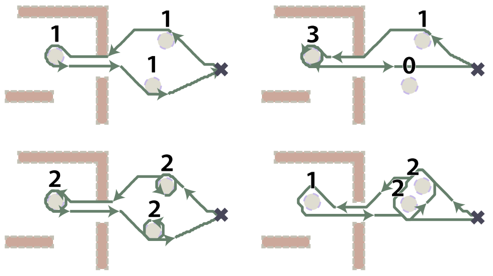
II. Related work

Our work can be considered a refinement of various recent methods involving planning trajectories subject to topological constraints. Most relevant are graph-based methods such as that of Bhattacharya et. al. [3], in which the state of a point-to-point navigation problem was augmented with a complex-valued state encoding information about path winding. Simplified variants using more direct encodings of the winding number were presented in [8] and [18]. These methods were 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 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



(a) Winding angle convention



(b) Winding constrained paths

Fig. 2. Illustration of winding angles (2a) and winding-constrained paths (2b). Fig. 2b shows winding-constrained paths in an environment with obstacles. Numbers indicate winding numbers of path with respect to the adjacent circular regions. Fig. 2b taken from [18], used with permission.

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 existing numerical methods to find such a path on a manifold. Although, recently developed heuristic-driven methods are significantly more efficient for this, intrinsically combinatorial, problem.

A. Problem formulation

In intuitive terms, our problem may be expressed simply as that of finding the minimum-cost continuous path in the plane while avoiding obstacles that also satisfies winding constraints. We assume the winding constraints are defined with respect to points located within obstacles 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 $[0\,,1] \to \mathbb{R}^2 \setminus \mathcal{O}$, in which $\mathcal{O}$ 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* $C : \mathbb{R}^2 \to \mathbb{R}^+$ (where $\mathbb{R}^+$ represents the positive reals). We then wish to find $\tilde{x}^*$, defined as the solution to the following optimization problem:

$$\begin{aligned} \tilde{x}^* \quad &= \quad \underset{\tilde{x}:[0\,,1]\to\mathbb{R}^2\setminus\mathcal{O}}{\arg\min} \int_{t=0}^{1} C(\tilde{x}(t))\|\dot{\tilde{x}}(t)\|dt \quad (1) \\ \text{subject to} \quad & \quad \tilde{x}(0) = s_x,\, \tilde{x}(1) = g_x \\ & \quad \tilde{\theta}(0) = s_\theta,\, \tilde{\theta}(1) = g_\theta \end{aligned}$$

Implicit in this optimization problem is another constraint coupling the optimized path $\tilde{x}$ in space to the dependent path of winding angles $\tilde{\theta} : [0\,,1] \to \mathbb{R}^N$; 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

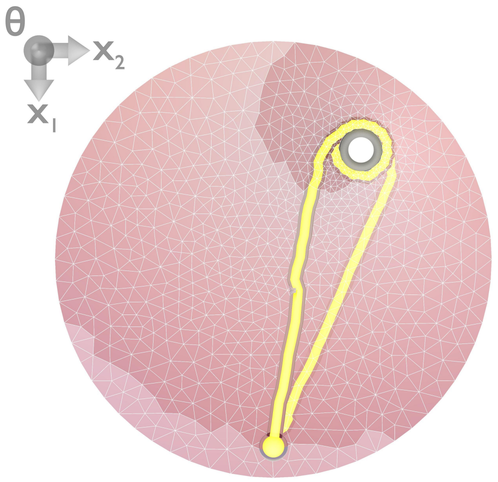
We first explore the geometry of the configuration space in the case that only one winding constraint is present. Fig. 3 shows a visualization of the configuration space for this problem. Since only one winding constraint is present, it suffices to keep track of one winding angle for the purpose of planning.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

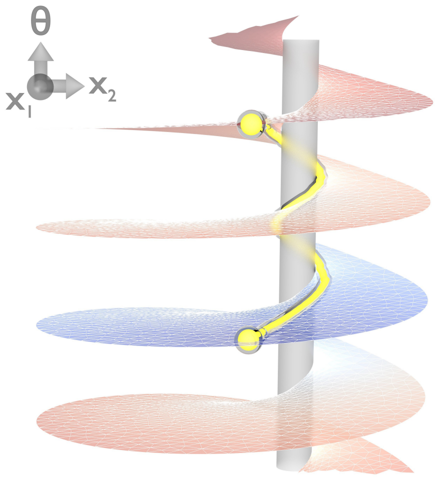
The general case appears significantly more complex, as the configuration space may contain winding information about multiple holes. It is perhaps counterintuitive, but it is possible to represent the configuration space as a surface embedded in $\mathbb{R}^2$ without self-intersections in the most general case.

The construction of this surface relies on a trick borrowed from [18], in which all of the of the winding angles θ_i are reversibly encoded into a single real value Θ. For example, Θ may be simply a linear combination of winding angles with coefficients that are equal to logarithms of distinct prime numbers z_i:

$$\Theta = \sum_i \theta_i \log z_i\,. \qquad (2)$$

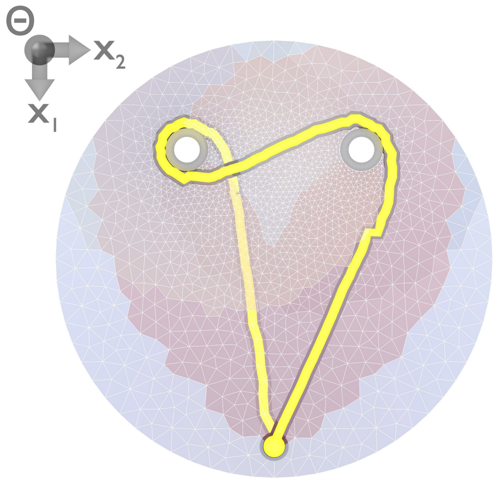


(a) Projection onto original planar domain

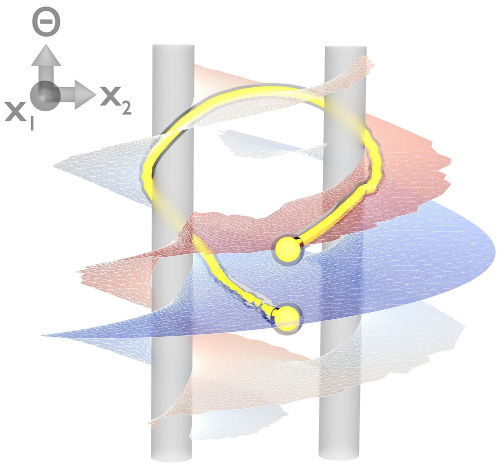


(b) Frontal view of winding-augmented configuration space

Fig. 3. Visualization of configuration space of planning with a single winding constraint. Fig. 3a shows the planar domain with a hole depicted as a white disc. The thick yellow tube depicts an optimal loop (computed by our method) constrained to wind twice around the hole. The path is constrained to begin and end at the yellow sphere. Fig. 3b shows a rendering of the *covering space* that constitutes the configuration space of the planning problem, along with the computed path in configuration space. Hue indicates cost-to-go function: redness of hue is proportional to distance to the goal in configuration space. Holes in configuration space are represented as gray cylinders. Small triangles show simplicial decomposition of surface.



(a) Projection onto original planar domain



(b) Frontal view of winding-augmented configuration space

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).

This idea is illustrated in Fig. 4b, in which we have visualized the configuration space with two winding constraints. Here we see that every feasible combination of winding angles at a given location x is represented by a unique point on a covering space of the twice-punctured plane.

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

Definition 3.1 (Joint position and winding dynamics):

$$\dot{\tilde{x}}(t) = \tilde{u}(t) \tag{3}$$

$$\dot{\tilde{\Theta}}(t) = \sum_{i=1}^{N} \dot{\tilde{\theta}}_i(\tilde{x}(t), \tilde{u}(t)) \log z_i \tag{4}$$

This definition allows us to draw a formal correspondence between the configuration space in (x, Θ) coordinates and a surface embedded in $\mathbb{R}^3$. The following theorem establishes this result.

Theorem 3.1: Assume $|\theta_i| < M, \forall i$ 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: The proof is technical and is omitted due to strict space requirements. ■

From the derivations above it follows that the optimization problem (2) is equivalent to the optimal control problem defined using the underactuated system (Definition 3.1) and the cost functional as in (2).

C. Hamilton-Jacobi-Bellman equation

The optimal control problem for underactuated system (Definition 3.1) results in the Hamilton-Jacobi-Bellman (HJB) Partial Differential Equation (PDE) for the cost-to-go function $V(x,\Theta) = \min_{\tilde{u}} L(\tilde{x}, \tilde{\Theta}, \tilde{u})$. The HJB is derived using Bellman's dynamic programming principle [1], and it reads:

$$0 = \min_{u \in U} \left\{ C(x)\|u\| + \nabla_x V(x,\Theta) \cdot u \right\}. \tag{5}$$

This equation above is accompanied with boundary conditions at g_x and g_Θ, such that $V(g_x, g_\Theta) = 0$. Clearly, once the robot is at the goal with correct winding angles, the cost-to-go is zero. Additionally, to guarantee a collision-free trajectory we require $V(x,\Theta) = \infty$ for all Θ if $x \in X_{\mathrm{obs}}$.

Once the optimal cost-to-go function is computed, a feedback policy function, $\pi : S \to U$, is given as a minimizer of the right-hand-side of (5). 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 the local projection. Thus, the cost-to-go function, V, and the optimal feedback policy, π, depend on Θ implicitly, as they may take different values at the same point x of the environment but different winding angles.

Generally, a closed-form solution does not exist for (5). Therefore, we must resort to numerical methods, which approximate the cost-to-go function. This leads to an approximated feedback policy, which, when integrated, produces an approximation of the shortest path that satisfies winding angle constraints.

IV. Numerical Algorithms

A. Simplicial Methods

Simplicial complexes are widely used in computational physics and numerical methods to approximate differential equations on smooth manifolds. Thus, motivated by Theorem 3.1, we discretize the configuration space using a two-dimensional simplicial complex, $(S_{\mathrm{d}}, \mathcal{T})$. Here, $S_{\mathrm{d}} = \{(x_i, \Theta_i) \in \mathbb{R}^3\}_{i=1}^N$ is a finite set of *verticies* sampled from the configuration space, and $\mathcal{T}$ is an *abstract* two-dimensional complex; see definitions in [20] and references therein. Intuitively, a two-dimensional simplicial complex is a triangular tessellation of the configuration space.

Define a *geometric representation*, S_T, of a simplex $T \in \mathcal{T}$ to be a convex hull of its nodal points, $\{x_i\}_{i \in T}$. Let $\hat{C}_T$ be a piecewise constant discretization of C, such that $\hat{C}_T$ is a constant on S_T, which is equal to the value of C at the simplex center. Let also $\hat{V}$ be a piecewise linear discretization of V, such that $\hat{V}(x,\Theta) = \sum_{i\in T} \alpha_i V(x_i, \Theta_i)$ for all $(x,\Theta) = \sum_{i \in T} \alpha_i (x_i, \Theta_i)$ in a geometric representation S_T. Function $\hat{V}$ is uniquely defined by its values at vertices of the simplicial complex, which we denote as $\hat{V}_i$ for simplicity. Considering (5) in the neighborhood of (x_i, Θ_i), we derive a discrete HJB equation

$$\hat{V}_i = \min_{T \in \mathrm{St}(i)} \inf_{x \in S_T} \left\{ \hat{C}_T \Big\| x_i - \sum_j \alpha_j x_j \Big\| + \sum_j \alpha_j \hat{V}_j \right\}, \tag{6}$$

in which $\mathrm{St}(i) = \{T \in \mathcal{T} \mid i \in T\}$ 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 $\hat{V}(g_x, g_\Theta) = 0$. Once (6) is solved, the approximate feedback policy is given as a minimizer of the right-hand-side in (6)

$$\hat{\pi}(x,\Theta) = \arg\min_{u \in U} \{ \hat{C}(x)\|u\| + \nabla_x \hat{V}(x,\Theta) \cdot u \}. \tag{7}$$

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

Equation (6) 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) [20]. 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

The SDA is an omnidirectional front propagation algorithm. Hence, it is unaware of extra information given in the case of a known initial robot position. To include this information and "focus" costly computations along the shortest path, we employ ideas from the A* algorithm [9]. Presented in [20] the Simplicial A* Algorithm (SAA) reduces the computation of the optimal cost-to-go function on a simplicial complex by considering an *admissible* and *consistent* heuristic, H, of the cost-to-come function. This algorithm is outlined below.

```
Input: Simplicial complex, $(S_{\mathrm{d}}, \mathcal{T})$, 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" vertices 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 $T \in \mathrm{St}(j)$ do
 6:         for all $i \in T \setminus \{j\}$ do
 7:             $(V^*, \pi^*) \leftarrow$ 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 [20] 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 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, Θ); i.e., from a starting state (x, Θ), 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 Θ 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

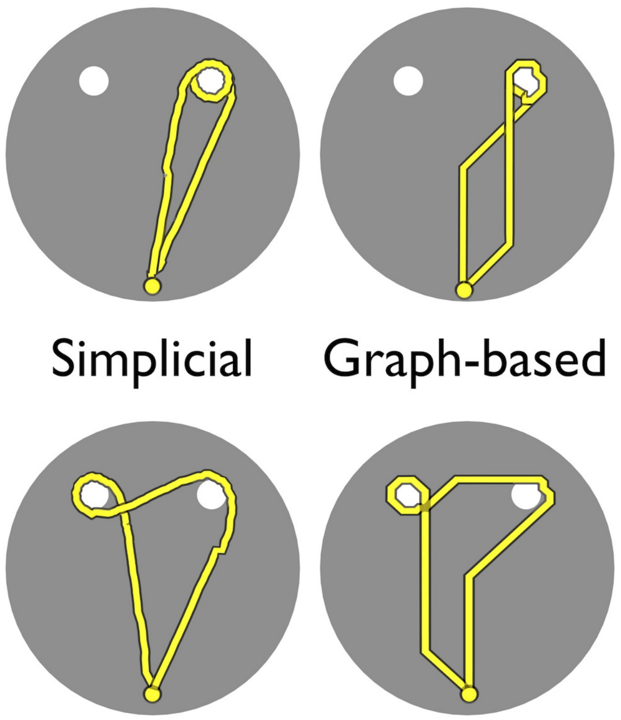
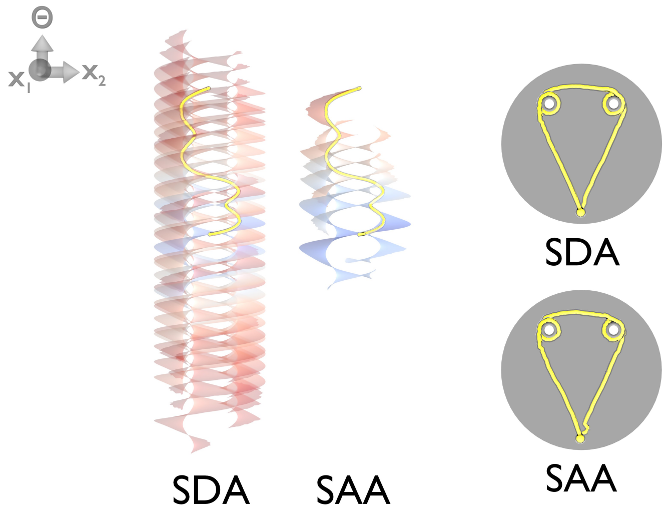
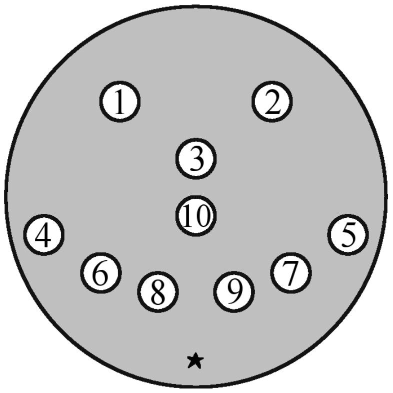


Fig. 5. Comparison of paths generated with our method (left images, labeled *simplicial*) against paths generated with the method of [18] (right images, labeled *graph-based*) for the example problems depicted in Fig. 3 and Fig. 4.

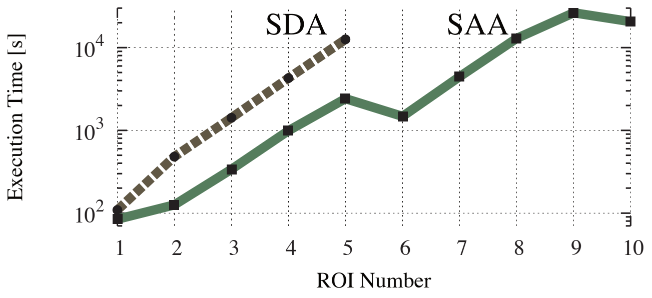


(a) Simplicial complexes (b) Paths

Fig. 6. Comparison of Simplicial Dijkstra (SDA) and Simplicial A* (SAA) algorithms for a problem specifying that the path wind twice in the clockwise direction around each hole. SDA generates a far larger simplicial complex approximation than SAA, but the found solutions paths are nearly identical.



(a) Experimental setup



(b) Scaling results

Fig. 7. Results of scaling experiments showing effect of varying number of winding constraints on runtime (note log scale), for both SDA and SAA methods. Fig. 7a shows test environment. Numbers indicate order in which winding constraints were introduced.

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

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—namely, 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

[1] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, March 1957.

[2] Sergei Bespamyatnikh. Computing homotopic shortest paths in the plane. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '03, pages 609–617, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[3] S. Bhattacharya, V. Kumar, and M. Likhachev. Search-based path planning with homotopy class constraints. In *Third Annual Symposium on Combinatorial Search*, 2010.

[4] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay Kumar. Identification and representation of homotopy classes of trajectories for search-based path planning in 3d. In *RSS*, 2011.

[5] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

[6] Alon Efrat, Stephen G. Kobourov, and Anna Lubiw. Computing homotopic shortest paths efficiently. *Computational Geometry*, 35(3):162 – 172, 2006.

[7] Dave Ferguson and Anthony Stentz. Field D*: An Interpolation-Based Path Planner and Replanner. In Sebastian Thrun, Rodney Brooks, and Hugh Durrant-Whyte, editors, *Robotics Research*, volume 28 of *Springer Tracts in Advanced Robotics*, chapter 22, pages 239–253. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[8] H. Gong, J. Sim, M. Likhachev, and J. Shi. Multi-hypothesis motion planning for visual object tracking. In *ICCV*. IEEE, 2011.

[9] Peter Hart, Nils Nilsson, and Bertram Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, February 1968.

[10] A. Hatcher. *Algebraic topology*. Cambridge University Press, 2005.

[11] J. Hershberger and J. Snoeyink. Computing minimum length paths of a given homotopy class. *Computational geometry*, 4(2):63–97, 1994.

[12] T. Igarashi and M. Stilman. Homotopic path planning on manifolds for cabled mobile robots. *WAFR*, pages 1–18, 2011.

[13] R. Kimmel and J. A. Sethian. Computing geodesic paths on manifolds. *PNAS*, 95(15):8431–8435, July 1998.

[14] J.R. Munkres. *Topology: a first course*. Prentice Hall, 1975.

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

[16] J. A. Sethian and A. Vladimirsky. Ordered Upwind Methods for Static Hamilton-Jacobi Equations: Theory and Algorithms. *SIAM Journal on Numerical Analysis*, 41(1):325–363, 2003.

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

[18] Paul Vernaza, Venkatraman Narayanan, and Maxim Likhachev. Efficiently finding optimal winding-constrained loops in the plane. In *RSS*, 2012.

[19] Dmitry S. Yershov and Steven M. LaValle. Simplicial Dijkstra and A* algorithms for optimal feedback planning. In *IROS*. IEEE, 2011.

[20] Dmitry S. Yershov and Steven M. LaValle. Simplicial Dijkstra and A* algorithms: From graphs to continuous spaces. *Advanced Robotics*, 26(17):2065–2085, October 2012.