Efficiently finding optimal winding-constrained loops in the plane

Paul Vernaza, Venkatraman Narayanan, and Maxim Likhachev
pvernaza@cs.cmu.edu, venkatrn@andrew.cmu.edu, maxim@cs.cmu.edu

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 19104

***Abstract*—We present a method to efficiently find winding-constrained loops in the plane that are optimal with respect to a minimum-cost objective and in the presence of obstacles. Our approach is similar to a typical graph-based search for an optimal path in the plane, but with an additional state variable that encodes information about path homotopy. Upon finding a loop, the value of this state corresponds to a line integral over the loop that indicates how many times it winds around each obstacle, enabling us to reduce the problem of finding paths satisfying winding constraints to that of searching for paths to suitable states in this augmented state space. We give an intuitive interpretation of the method based on fluid mechanics and show how this yields a way to perform the necessary calculations efficiently. Results are given in which we use our method to find optimal routes for autonomous surveillance and intruder containment.**

I. Introduction

The subject of this work is finding optimal constrained loops in the plane. More specifically, out of all paths starting and ending at a specific location in the plane and satisfying certain *winding* constraints, we would like to find one such path that minimizes a given location-dependent cost accumulated along the path, while also avoiding some regions entirely.

Figure 1 depicts a practical situation in which this type of problem arises. Here, an unmanned aerial vehicle (UAV) is tasked with flying a surveillance mission to photograph certain regions of interest (ROI) located within hostile territory. The UAV must find a route that begins and ends at its home base. It must additionally fly a route that minimizes a given cost functional, determined by distance traveled and proximity to hostile radar installations, while entirely avoiding certain high-traffic regions deemed too risky to traverse. Of prime concern is the need to photograph the ROI from all perspectives. Formally, we can ensure this by requiring that the generated path satisfy topological *winding* constraints with respect to the ROI—if the UAV winds at least once around each ROI, it will be able to view each ROI from every angle.

The rest of this paper is organized as follows. First, we give an intuitive description of our approach, followed by a brief discussion of the relation of our work to other methods. An in-depth description of technical details of our method follows in Section IV. Experimental results are given in Section V, followed by discussion.

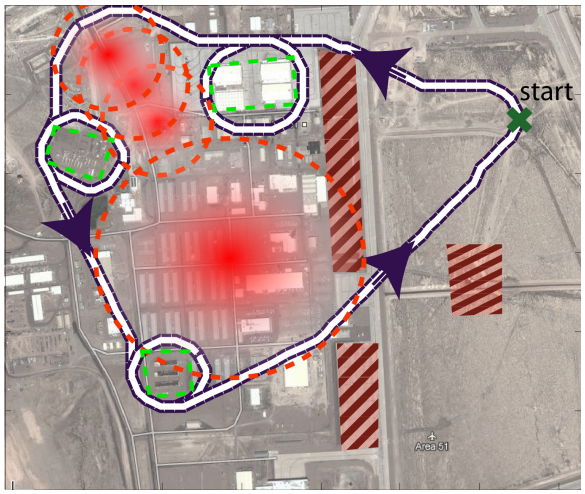


Fig. 1: An optimal plan (solid line) for a hypothetical UAV surveillance mission. The UAV is constrained to take off and return to a particular location, *winding* twice around each of three regions of interest. Regions of interest are designated by polygonal dashed lines. Red, striped polygons denote high-traffic regions that the UAV must avoid. Circular dashed lines denote location and ranges of radar installations, with radar power decreasing with distance to center.

II. Method Overview

The basic idea of our method is very simple and best motivated by a fluid analogy. Suppose we wish to find a planar loop that encloses some set of regions. We imagine that within each region is a source that produces fluid at a certain rate. By the divergence theorem (illustrated in Fig. 2), we know that the net rate at which fluid escapes any loop is equal to the rate at which it is produced by sources enclosed by the loop. Therefore, we can find a loop enclosing the desired regions by searching for a loop through which fluid escapes at a rate equal to the sum of the rates assigned to those regions. So long as there are no two sets of regions whose rates sum to the same quantity, we can be assured that any such loop must contain only the desired regions.

We can find such a loop via a straightforward modification of any standard graph-based method for navigation in the

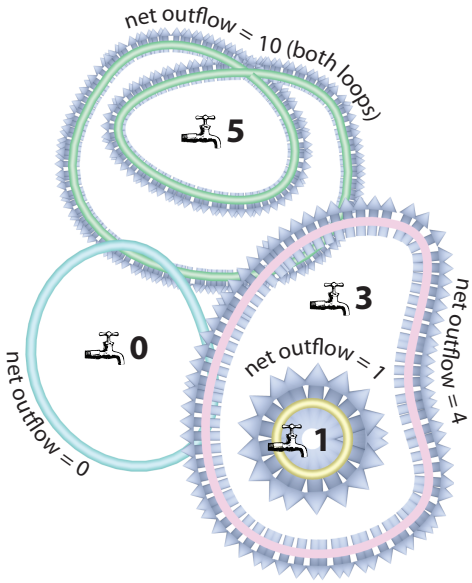


Fig. 2: Illustration of divergence theorem: total flow rate exiting the boundary of any region is equal to the sum of source flow rates contained within, even in the presence of other sources. Locations of flow sources are indicated by faucet icons, with their flow rates indicated by numbers adjacent to these icons.

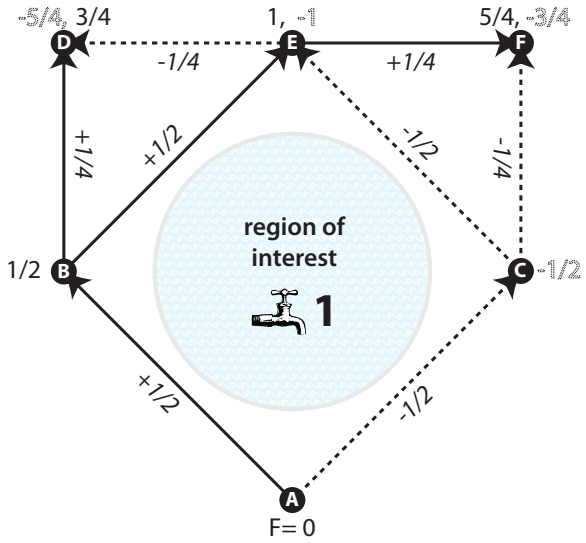


Fig. 3: Illustration of homotopy-augmented search. Shown is a 2x2 grid on which a graph search is performed, with a single obstacle in the center. Graph search begins at point A (marked $F = 0$). Directed edges are marked with signed F-value increments (in italics). Other values denote F-values at grid vertices associated with one or more paths discovered during search. Solid lines show paths found passing the obstacle to the left, while dashed lines show paths found passing the obstacle to the right

plane. This is accomplished by treating the rate at which fluid passes through a path as a dependent variable whose state is tracked along with the usual state necessary for navigation, such as position and orientation. Any given state of the vehicle can therefore be associated with any number of different flow values, depending on how the search reached that state. When the search visits the goal configuration, we can check the flow value to verify whether the desired regions were enclosed; if they have been, then we are assured that the loop thus found is optimal with respect to all other loops satisfying our constraints, as long as the search algorithm employed is *admissible*.

A. Example with one region

Fig. 3 gives a simple illustration of how our method might proceed given just one region of interest around which to loop. Specifically, we consider the problem of finding an optimal path that begins at the *flow-augmented state* $((x, y), F) = (A, 0)$ and winds around the obstacle one or more times before returning to the location A. We will refer to the flow state F as the F-value.

The figure shows a number of possible actions that the search could take, for illustrative purposes. Each arc is marked by the quantity by which the F-value is incremented traversing the arc in the direction shown, given that the region of interest contains a source of rate one. Each location is marked by the F-values found at those locations by exploring different illustrated paths. For instance, the search begins by expanding A, generating B with $F = 1/2$, and generating C with $F = -1/2$. Next, the search expands B, generating D with $F = 3/4$ and E with $F = 1$, followed by a similar expansion of C.

At this point, we note that E has been reached via two (non-homotopic) paths, each yielding a distinct F-value. Therefore, we next expand the location E twice—once for the state $(E, 1)$ and once for the state $(E, -1)$. If we continue this process, the reader may verify that we will eventually expand the state $(A, 2)$ and/or $(A, -2)$, though this is not explicitly illustrated for purposes of clarity. Note that this will require traversing arcs in directions contrary to those illustrated, in which case the arc's F value is negated before adding it to the state's F-value.

Upon expanding the state $(A, 2)$, it is apparent that the search has found a path winding clockwise around the obstacle exactly once. Furthermore, it may be verified that any other path that winds once, clockwise, around the obstacle, also has $F = 2$. However, if we expand the states in an *admissible ordering*, such as that provided by A* [9], we are guaranteed that the first path via which we have expanded $(A, 2)$, is the optimal one to that state. Since all paths starting at A, winding clockwise, once, around the obstacle, arrive at $(A, 2)$, we can in that case conclude that this path is one that achieves the minimum cost out of all such paths.

We finally observe that any loop winding around the obstacle k times arrives at the state $(A, \pm 2k)$ for $k > 0$, with sign determined by the winding direction. If the loop does not

wind around the obstacle, however, the path arrives at the state $(A, 0)$. The converse statements are also true.

B. Winding around multiple regions

In the case that there are multiple regions of interest, we proceed as previously mentioned, placing a fluid source inside each region and computing F-values via superposition. We may then specify how many times the path should wind around each region, where *winding* is defined in the topological sense [10]. Assuming we assign to each region a fluid source of rate $\log z_i$, where z_i is the ith prime number, a loop satisfying the winding constraints may be found by searching for a path to a state with

$$\exp F = \Pi_i z_i^{W_i}, \tag{1}$$

where W_i is the winding number associated with the ith region. The fundamental theorem of arithmetic implies that this constitutes an invertible map between winding numbers and F-values.

III. Related Work

Our work was inspired by the work of Bhattacharya et. al. [3, 4], who first proposed the general idea described in Section II of performing graph search for navigation with a state vector augmented by homotopy information. Our work differs principally in two ways. First, we derive a version of homotopy-augmented graph search that is suitable for finding optimal loops with arbitrary winding constraints. Furthermore, we employ a real-valued encoding of homotopy information, as opposed to the complex-valued encoding derived in [3].

Although it is not the focus of this work, we note that the real-valued encoding of homotopy information derived here may also be applied to the problem described in [3], which might be characterized as point-to-point planning in 2D with homology constraints on the generated paths. In addition to having an intuitive interpretation in terms of fluids, applying the encoding we derive here would yield a method arguably simpler to implement.

Also noteworthy is recent work in computer vision that employs planning with winding angles in order to perform tracking with occlusion [8]. Planning in this way is essentially equivalent to the method described here, a fact that we show formally in Sec. IV-C. Additionally, we give an intuitive fluid-based interpretation of the method, derive an appropriate heuristic, and apply the method to surveillance and optimal confinement problems.

The problem of loop planning discussed here also bears some similarity to several problems in computational geometry. In the geometric knapsack problem [1], the objective is to find a simple loop enclosing discrete reward locations of varying reward, such that the enclosed reward minus the length of the loop, is maximized. Arkin et al. [1] give algorithms to solve certain variants of this problem in time polynomial in the number of vertices of the polygons given as input. Our method is similar if the reward regions are thought of as regions around which the path should wind at least once. However, our method appears to be more flexible in a variety of ways, due to our use of graph-based search. We need not assume the environment is polygonal, nor do we assume the cost of the path is uniform across free space, or that the vehicle dynamics are trivial, as in [1]. Moreover, we can leverage problem-specific heuristics in the context of A* in order to accelerate the search. A final difference between our work and that of [1] is that we are able to find solutions with arbitrary winding constraints.

As surveillance constitutes a principal application of our method, our method is related to another problem from computational geometry knows as the *optimum watchman route* problem. This problem is defined as that of generating a path of minimum length through an environment such that every point in the environment is visible from some point along the path. We motivate the UAV surveillance problem considered here from an aim of rendering each point along the ROI visible, but the nature of our solutions is such that this would be guaranteed only in certain cases, such as the case where ROI are flat enough to guarantee that no ROI occludes any other, or the case where the ROI are spaced sufficiently such that winding twice around each ROI is sufficient to guarantee full visibiity. Therefore, it is unlikely that optimum watchman routes are equivalent to optimum winding-constrained loops.

UAV planning in itself has been the focus of a significant amount of work in engineering disciplines. Typical goals of these approaches include cooperative planning, enforcing non-holonomic constraints, and planning in real-time [6, 5, 2]. Other work focuses on ensuring geometric visibility in urban environments [7]. However, it appears that comparatively little work has focused on planning optimal loop paths for UAVs. An exception is the work of [11], who devised a suboptimal algorithm to find looping paths visiting a number of ROI while satisfying path curvature constraints. This method is again less flexible than ours in that we allow an arbitrary spatially-dependent cost function to take into account factors such as spatially-varying risk.

IV. Technical Details

We now clarify some of the technical points of the method presented in Sec. II.

A. Graph construction

An informal description of our graph construction was given in Section II. That description may be formalized in the following way. First, we assume we are given a description of the graph corresponding to a point-to-point navigation problem in the plane, consisting of a state space $\mathcal{X}$ and a successor function succ : $\mathcal{X} \to 2^{\mathcal{X}}$, where $2^{\mathcal{X}}$ is the power set of $\mathcal{X}$. Typical examples of $\mathcal{X}$ include $\mathcal{X} = \mathbb{R}^2$ and $\mathcal{X} = \mathbb{R}^2 \times S^1$, the latter corresponding to navigation with heading dependence.

We then define a new state space $\mathcal{X}' = \mathcal{X} \times \mathbb{R}$, with elements denoted by (x, f). Let $\rho : \mathcal{X} \times \mathcal{X} \to (I \to \mathbb{R}^2)$ denote the function that generates the Cartesian path taken from a node in the original graph to a successor. succ$'$ is then defined in the following way:

$$\text{succ}' : (x, f) \mapsto \{(x', f') \mid x' \in \text{succ}(x), f' = f + F(\rho(x, x'))\},$$

where $F : (I \to \mathbb{R}^2) \to \mathbb{R}$ is a line integral that will be defined formally in the next section.

The start state for graph search is defined by $(x_0, 0)$, where x_0 is the start state in the pre-augmented graph. The goal state is defined as $(x_0, \sum_i W_i \log z_i)$, where W_i is the desired winding number of the solution path around the ith region.

B. Line integral construction

Aspects of the line integral referred to in the last section are now detailed. First, we note that a suitable flow field is given by

$$V(x) = \sum_i \frac{r_i}{2\pi} \frac{x - o_i}{\|x - o_i\|^2}, \tag{2}$$

where $o_i \in \mathbb{R}^2$ is an arbitrary source location inside the ith region for which we are given a winding constraint, and $r_i = \log z_i$ is the source rate assigned to the ith source. This may be established by considering, for example, the case where o_i is located at the center of a circle and devising an integrand that will integrate to r_i. Denoting by $\hat{n}(s)$ the normal to $\tilde{x}$ at s, we define

$$F(\tilde{x}) = \int V(\tilde{x}(s)) \cdot \hat{n}(s)\, ds. \tag{3}$$

In order to compute the value by which F increments along an edge, we must evaluate this integral with respect to the path taken by the edge. Assuming this path is a straight line, the integral can be computed analytically. The integral is simplified considerably by taking into consideration its rotational, translational, and scaling invariances, as depicted in Fig. 4. By further applying linearity, we may therefore reduce the general case to the line integral of V over $(0,0) \to (0,1)$ due to a single obstacle of rate r located at position $o = (o_x, o_y)$. Assuming the segment normal is given by $\hat{n} = \begin{pmatrix} -1 & 0 \end{pmatrix}$ the absolute value of the line integral is in this case given by

$$\int_{s=0}^{1} V(x) \cdot \hat{n}\, ds = \frac{r}{2\pi} \int_{y=0}^{1} -\frac{-o_x}{o_x^2 + (y - o_y)^2} dy \tag{4}$$

$$= \frac{r}{2\pi} \left(\arctan \frac{1 - o_y}{o_x} - \arctan \frac{-o_y}{o_x} \right), \tag{5}$$

which is equal to $r\theta/(2\pi)$, with θ as defined in Fig. 4. The general case may then be derived via similar triangles and superposition. For a line segment with generic endpoints p_0 and p_1, this yields

$$F = \sum_i s_i \frac{r_i}{2\pi} \arccos \left\langle \frac{p_1 - o_i}{\|p_1 - o_i\|}, \frac{p_0 - o_i}{\|p_0 - o_i\|} \right\rangle, \tag{6}$$

where the sign $s_i \in \{-1, 1\}$ is determined by whether o_i falls on the left or right side of the directed segment $p_0 \to p_1$, according to the desired convention.

C. Equivalence with winding-vector approach

Summing Equation (6) along a path reveals the F-value to be a linear projection of the vector of winding angles associated with the path around each region; i.e., $F = \langle w, \theta \rangle$, where θ is a vector of winding angles, and w is a vector

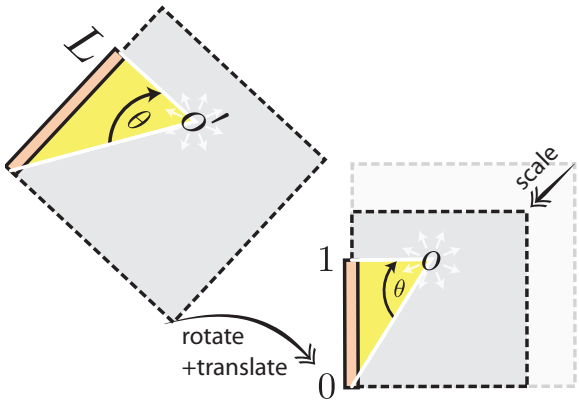


Fig. 4: Illustration of symmetries in line integration of flow field: the flux through the segment L due to o' is equal to the flux through the segment $(0,0) \to (0,1)$ due to o.

with $w_i = \log z_i/(2\pi)$. The correctness of Eq. (1) follows immediately from this observation. This raises the question of whether an equivalent method may be obtained by augmenting the state vector with a vector of winding angles as opposed to the single F-value. Here we show that the two methods are indeed equivalent, in the sense that the entire vector of winding angles can be uniquely recovered from *any* F-value.

The key observation is that each valid winding angle at each location in the plane can be expressed in the form $\theta_i = \bar{\theta}_i + 2\pi k_i$, where $k_i \in \mathbb{Z}$ and $\bar{\theta}_i$ is a location-dependent offset angle. We then observe that

$$\exp F = \exp \sum_i \log z_i \left(k_i + \frac{1}{2\pi} \bar{\theta}_i \right), \tag{7}$$

which implies that

$$\exp \left(F - \sum_i \frac{\log z_i}{2\pi} \bar{\theta}_i \right) = \Pi_i z_i^{k_i}. \tag{8}$$

Therefore, given F, the winding angles θ_i can be obtained by finding the prime factors of the left-hand side of the previous expression.

An interesting geometric interpretation of this fact is that at each location, the winding angles are determined by the intersection of the hyperplane $F = \langle w, \theta \rangle$ and the discrete, infinite lattice of valid winding angles. The slopes of this hyperplane are ratios of logs of prime numbers, and are therefore irrational; consequently, the hyperplane can intersect the infinite lattice in at most one location. This is illustrated in Fig. 5.

A practical consequence of this analysis concerns the memory efficiency of the method. Given many winding constraints, it is clearly inefficient to take the naive approach of storing each winding angle as an individual floating-point value, since the set of valid winding angles for each location is a discrete set. Storing a single F-value as a floating-point value constitutes a memory-efficient alternative to this approach. Alternatively, one may store a single floating-point offset angle

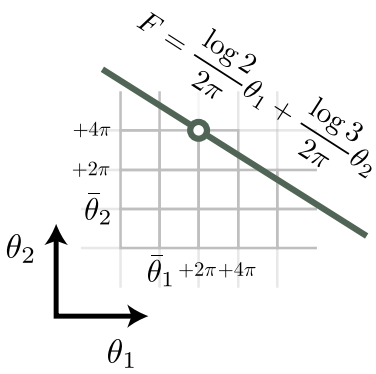


Fig. 5: Illustration of invertibility of map from winding angles to F-values for an example with two regions of interest. The set of valid winding angles (θ_1, θ_2) consists of the set of lattice points (intersections of gray lines). The only valid pair of winding angles that also satisfies the F equation is the unique intersection of the pictured line with the lattice (circled). The uniqueness of the intersection point follows from the irrationality of the slope $-\log 2/\log 3$.

vector $\bar{\theta}$ per location in addition to a vector of integer k_i per state; for any reasonable problem, only a few bits per state would be necessary to represent each k_i.

D. Search heuristic

The combinatorial nature of winding constraints necessitates a careful choice of search strategy. For this reason, we employ A* search with the heuristic illustrated in Fig. 6. Given a query state for which the heuristic is to be computed, we calculate, of the regions with winding constraints, which regions would have their constraints satisfied if the loop were closed with a straight line to the goal. For each of the remaining regions with non-zero winding constraints, we compute the length of the minimum-length path beginning at the query state, touching the region, and returning to the goal location; this constitutes the minimum-length excursion necessary to satisfy that region's winding constraint. Of all the distances so computed, we choose the greatest to be the value of the heuristic.

V. Experiments

We implemented the method and applied it to several test scenarios described here, with the aim of understanding the qualitative characteristics of planning with winding constraints, the applicability of the method to realistic problems, and the method's computational efficiency. For simplicity, the examples presented here only consider curves with positive winding, though constraints with negative winding present no obstacle to the method.

A. Synthetic environment

Fig. 7 shows the result of a simple synthetic experiment showing optimal plans generated by our method winding around three regions (referred henceforth as ROI) in varying ways, and in the presence of obstacles. The purpose of this

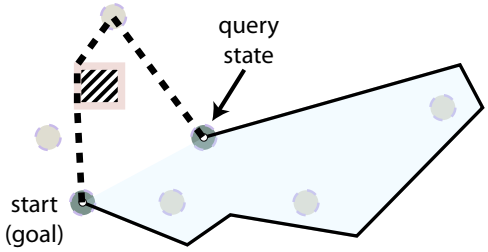


Fig. 6: Illustration of heuristic. *Query state* indicates state for which heuristic is to be evaluated. Solid line shows a path that may have been taken to this state. Regions with positive winding constraints are depicted as solid circles. Striped area is an obstacle. Shaded area encloses regions whose winding constraints would be satisfied if the loop were closed with a straight line to the goal. Dashed line shows the maximum-length excursion to reach any winding-constrained region not in the shaded area. The value of the heuristic is the length of the dashed line.

experiment was to examine the qualitative aspects of paths with different winding constraints.

In the simplest case, the path is constrained to wind around each ROI exactly once. Since we simulated no vehicle dynamics in this experiment, the optimal solutions must always be Jordan [10] in this case, as depicted. Paths constrained to wind twice around some ROI tend to consist of a large loop winding once around each obstacle with offshoots that loop around each ROI once more. However, if any pair of ROI are sufficiently close, the optimal path may contain a Jordan subloop containing both, as depicted in the lower-right portion of the figure. From an applications perspective, this type of behavior may be desirable in the case of surveillance with limited sensing range. In this case, it may be sufficient to wind around groups of ROI (instead of winding around each individually), provided that each group is contained within a circle of radius smaller than the agent's sensing range.

Note also that it is possible to specify a zero winding number for certain ROI. In the case that all the windings are either zero or one, the ROI with winding one must be contained within the region enclosed by the loop (which must be Jordan), while the ROI with winding zero must fall outside the loop.

B. UAV surveillance

We now return to the UAV surveillance problem mentioned in the introduction. For these experiments, we simulated hypothetical UAV missions by annotating aerial imagery of military bases with ROI, hypothetical radar installations, and regions of excessive risk to be avoided entirely. Radar installations were modeled as overlapping ellipses, each containing a cost attaining its maximum value at the center of the ellipse and diminishing gradually to zero at the boundary of the ellipse. Our objective was to find minimum-cost loops subject to winding constraints and entirely avoiding regions of excessive risk. We approximated nontrivial dynamics for the vehicle by planning

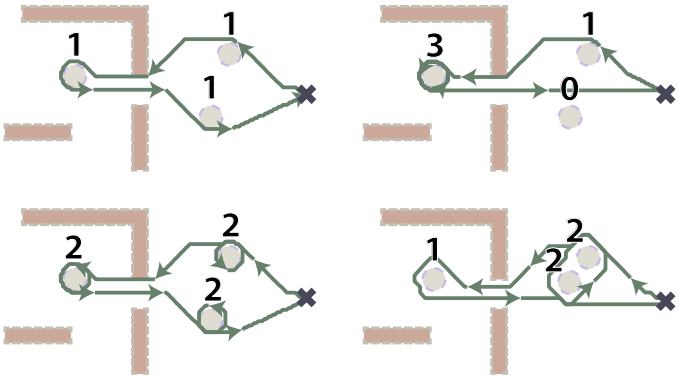


Fig. 7: Demonstration showing loops found by the method to satisfy different winding constraints. Numbers indicate winding number constraints enforced for circular regions located directly beneath. Obstacles depicted by rectangular regions.

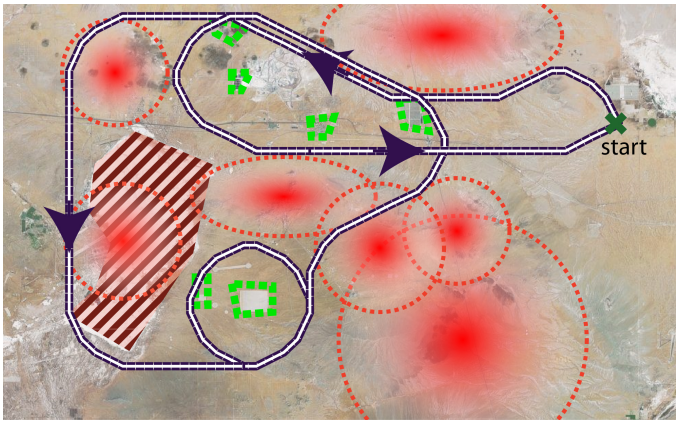


Fig. 8: Result of path planning for hypothetical UAV surveillance mission. UAV is constrained to wind exactly twice around each ROI. Annotations are as described in Fig. 1.

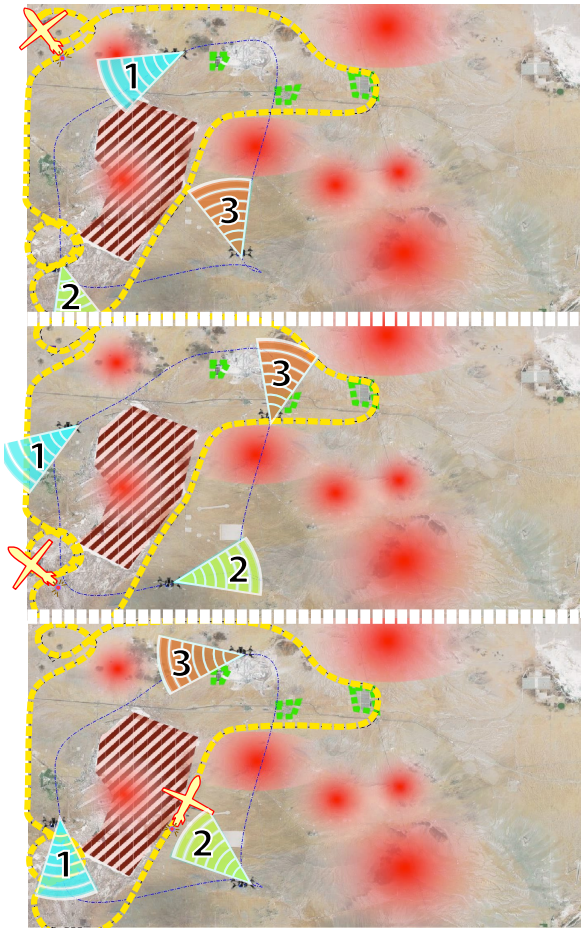


Fig. 9: Results of path planning for UAV surveillance with time-varying cost function. Curved, dashed line shows planned path. Colored, striped wedges show fields of view of enemy patrollers. Three frames are shown, each showing the position of the UAV and patrollers at a particular point in time.

for its orientation as well, limiting the maximum change in orientation between straight segments to 22.5 degrees.

Figure 1 shows the result of finding an optimal path winding around each ROI exactly twice. The optimal path in this case has the previously discussed characteristic of having a large outer loop with smaller loops branching off of it. It is observed that the path generally stays just outside the region of radar visibility, straying inside only briefly in order to wind around the ROI. As expected, the path also never wanders inside the forbidden regions.

Figure 8 shows a result obtained in another scenario. Again, the path was constrained to wind twice around each ROI. An interesting observation here is that upon visiting the last ROI, the UAV has a choice of either winding around the ROI and roughly retracing its path in the opposite direction, or penetrating radar range briefly in order to return to its home base. We observe that the optimal path indeed penetrates radar range via a short path near the boundaries of two overlapping radar installations before completing the loop.

We additionally attempted to generate plans for a scenario in which enemy patrol units were present. These were modeled by augmenting the state with a time variable; high costs were assigned to those positions and times representing states in which the UAV could be observed by an enemy patrol unit. Fig. 9 shows an example result. The optimal path successfully allows the UAV to evade the patrol units. An interesting feature of the plan is a small loop that effectively slows the UAV in order to avoid detection by patrol unit 2.

C. Intruder confinement

We finally studied the application of our method to a problem we will refer to as *intruder confinement*. This problem consists of finding an optimal way to confine several intruders in a maze-like environment, subject to the constraint that innocent bystanders should not be confined. Confinement is achieved by surrounding the intruders with a team of robots that deploy from a central location. We assume that the robots have limited communication range, such that we can define a *connectivity graph* having an edge between two robots iff. they are within communication range of each other. In order to

ensure that the robots can coordinate, this graph should remain *connected* at all times.

A feasible deployment strategy may be obtained by finding a loop that winds around each intruder exactly once without winding around a bystander. Robots may then incrementally deploy along the loop at intervals smaller than their communication radius in order to surround the intruders while maintaining full connectivity and allowing bystanders to escape.

The results shown in Fig. 10 demonstrate that the method is able to find optimal enclosures in highly irregular, maze-like indoor environments of varying topologies. As was expected, we observed that the complexity of planning did not scale significantly with the complexity of the environment, allowing us to solve problems such as those depicted. However, the complexity was significantly affected by the number of winding constraints imposed, as will be discussed shortly.

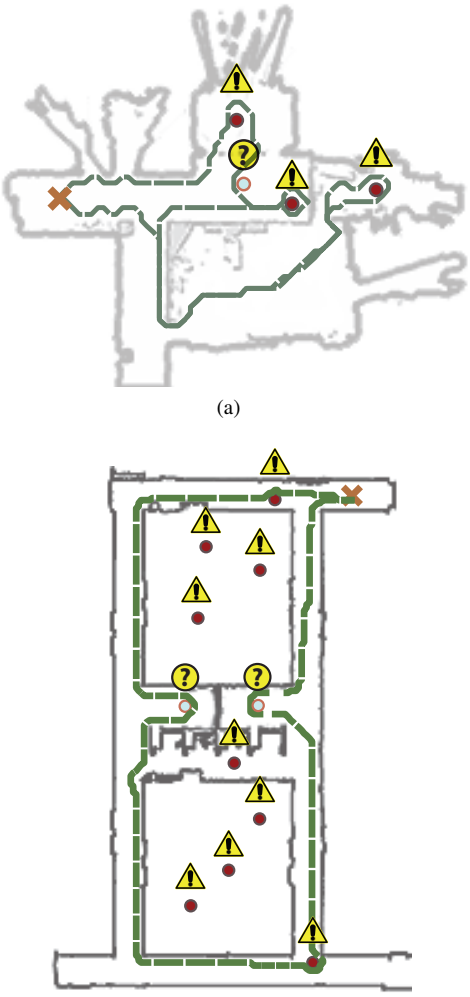
D. Computational efficiency

In order to study the computational efficiency of the method, we applied it to a synthetic example where we varied just the number of regions with winding constraints. The experimental scenario is depicted in Fig. 11. In the nth trial, we used our method to find a loop around the regions labeled $1, 2, \ldots n$, winding once in the positive direction around each. This was conducted both using the heuristic described in Sec. IV-D—which we will refer to as the *loop heuristic*—and using the Euclidean distance-to-goal heuristic.

Fig. 11b shows a clearly asymptotic exponential scaling in the run time as a function of the number of winding constraints for both heuristics. The loop heuristic, however, exhibits a significantly more shallow slope in the log plot; for 10 constraints, the Euclidean distance heuristic is slower by a factor of 10. An interesting observation is that the performance of the loop heuristic was relatively insensitive to the number of constraints up to about six, at which point planning times increased significantly. For a very small number of constraints, the Euclidean distance heuristic outperformed the loop heuristic by a large margin, which we attribute to the smaller overhead of the Euclidean heuristic.

The reason for the observed exponential scaling is that the search ultimately begins to explore many different combinations of windings around different obstacles. As we increase the number of constraints, the number of plausible winding states (and by extension, the open list) grows exponentially. The loop heuristic was able to compensate for this effect somewhat, but not completely. We also observed in experimens with constraints winding multiple times around each region, that the performance of the loop heuristic deteriorated. This is to be expected, as it does not take into account the effect of multiple windings.

We are currently investigating different avenues to enable scaling to yet larger numbers of winding constraints. One simple enhancement would be to prune the search when winding angles become too large (a similar strategy was used in [8]), at the expense of losing the completeness guarantee of the search. Also, as noted in Section III, certain variants of the



(a)

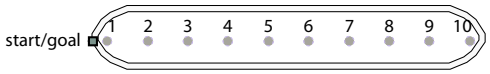
(b)

Fig. 10: Results of intruder confinement experiments. Locations of intruders are marked by circles below exclamation symbols. Locations of bystanders are marked by circles below question marks. Paths generated by the method are shown superimposed on floor plan of environment.

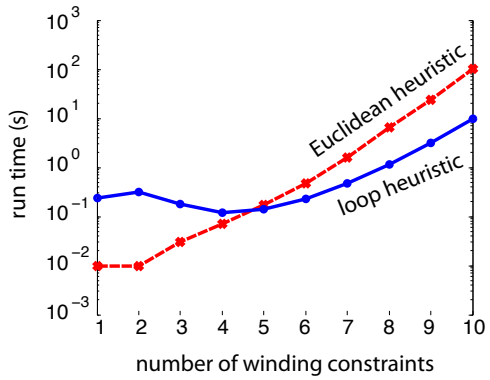
geometric knapsack problem in computational geometry may be considered relaxations of the winding-constrained planning problem, and may be solved efficiently. An admissible heuristic might therefore be obtained as the solution to such a problem. Alternatively, it is possible to construct an admissible heuristic as a graph search problem on a vastly reduced graph that assumes a uniform cost function. We leave implementation of these ideas as future work.

VI. Conclusion

We have demonstrated a method for efficiently finding optimal loops in the plane subject to winding constraints. The method accomplishes this by collapsing together all paths sharing a common F-value, which encodes homotopy-type information about the paths. The F-value is constructed in such a way as to reversibly encode winding data when a



(a) Experimental setup



(b) Run time scaling

Fig. 11: Synthetic experiment to determine empirical scaling of computational complexity as a function of number of winding constraints. Fig. 11a shows experimental setup with point obstacles around which winding constraints were imposed, along with order in which these constraints were introduced and the solution path for 10 obstacles. Fig. 11b shows the time to find the optimal solution versus the number of winding constraints (note log scale), with and without the heuristic described in Sec. IV-D (labeled *loop heuristic*).

loop is completed, allowing us to reduce a search for cyclic, winding-constrained paths to a search for an acyclic path to a specific state encoding the desired winding. Computation of the F-value additionally has an intuitive fluid-based interpretation and may be performed efficiently. Finally, our method leverages standard graph-based search methods to achieve flexibility in problem representation, vehicle dynamics, and selection of domain-specific admissible heuristics to accelerate the search process.

A particular challenge for the method as implemented up to this point is the case where many winding constraints are to be enforced. In the near-term, we anticipate that employing straightforward heuristics (as described in Section V-D) will greatly enhance the ability of the method to solve problems of this type.

An additional area of future research is using our method as a stepping stone to solving problems with constraints related to, but stronger than winding. For instance, in the UAV surveillance problem, we may wish to enforce the constraint that the UAV fully observe one location before proceeding to another. We are currently investigating how this type of constraint might be enforced by augmenting the state vector with state that keeps track of such ordering information. Though the variations are endless, we believe that winding-constrained planning yields fundamental insight that will prove useful in a variety of related endeavors.

Acknowledgments

This work was supported by ONR DR-IRIS MURI grant #N00014-09-1-1052 and ONR ANTIDOTE MURI grant #N00014-09-1-1031.

References

[1] E.M. Arkin, S. Khuller, and J.S.B. Mitchell. Geometric knapsack problems. *Algorithmica*, 10(5):399–427, 1993. URL http://www.springerlink.com/content/g007w81p153h3326/.

[2] J.S. Bellingham, M. Tillerson, M. Alighanbari, and J.P. How. Cooperative path planning for multiple UAVs in dynamic and uncertain environments. In *IEEE Conferenece on Decision and Control*, volume 3, dec. 2002.

[3] S. Bhattacharya, V. Kumar, and M. Likhachev. Search-based path planning with homotopy class constraints. In *Third Annual Symposium on Combinatorial Search*, 2010. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1920.

[4] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay Kumar. Identification and representation of homotopy classes of trajectories for search-based path planning in 3d. In *Proceedings of Robotics: Science and Systems*, 27-30 June 2011. URL https://www.aaai.org/ocs/index.php/SOCS/SOCS10/paper/view/2089/0.

[5] S.A. Bortoff. Path planning for UAVs. In *American Control Conference*, volume 1, pages 364 –368 vol.1, sep 2000. doi: 10.1109/ACC.2000.878915.

[6] P.R. Chandler, M. Pachter, and S. Rasmussen. UAV cooperative control. In *American Control Conference*, volume 1, pages 50 –55 vol.1, 2001. doi: 10.1109/ACC.2001.945512.

[7] Peng Cheng, J. Keller, and V. Kumar. Time-optimal UAV trajectory planning for 3d urban structure coverage. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 2750 –2757, sept. 2008. doi: 10.1109/IROS.2008.4650988.

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

[9] P.E. Hart, N.J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100 –107, july 1968. ISSN 0536-1567. doi: 10.1109/TSSC.1968.300136. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4082128.

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

[11] Zhijun Tang and U. Ozguner. Motion planning for multitarget surveillance with mobile sensor agents. *Robotics, IEEE Transactions on*, 21(5):898 – 908, oct. 2005. ISSN 1552-3098. doi: 10.1109/TRO.2005.847567.