

Sampling-Based Path Planning for a Visual Reconnaissance UAV*

Karl J. Obermeyer[†]

University of California at Santa Barbara, Santa Barbara, CA 93106

Paul Oberlin[‡] and Swaroop Darbha[§]

Texas A&M University, College Station, TX 77843

This article considers a path planning problem for a single fixed-wing aircraft performing a reconnaissance mission using EO (Electro-Optical) camera(s). A mathematical formulation of the general aircraft visual reconnaissance problem for static ground targets in terrain is given and it is shown, under simplifying assumptions, that it can be reduced to a *PVDTSP* (*Polygon-Visiting Dubins Traveling Salesman Problem*), a variation of the famous TSP (Traveling Salesman Problem). Two algorithms for solving the PVDTSP are developed. They fall into the class of algorithms known as *sampling-based roadmap methods* because they operate by sampling a finite set of points from a continuous state space in order to reduce a continuous motion planning problem to planning on a finite discrete graph called a *roadmap*. Under certain technical assumptions, the algorithms are *resolution complete*, which means the solution returned provably converges to a global optimum as the number of samples grows, i.e., as the resolution of the roadmap becomes finer. The first algorithm is resolution complete under slightly milder assumptions, but the second algorithm achieves faster computation times by a novel roadmap construction. Numerical experiments indicate that, for up to about 20 targets both algorithms deliver very good solutions suitably quickly for online purposes. Additionally, the algorithms allow trade-off of computation time for solution quality and are shown extensible to handle wind, airspace constraints, any vehicle dynamics, and open-path (vs. closed-tour) problems.

I. Introduction

UAVs (Unmanned Air Vehicles) are increasingly being used for both civilian and military applications such as environmental monitoring, geological survey, surveillance, reconnaissance, and search and rescue.^{1,2} Good control and planning algorithms are a key component of UAV technology because they can increase operational capabilities while reducing risk, costs, and operator workloads. In this article we present a novel path planning algorithm for a single fixed-wing aircraft performing a reconnaissance mission using EO (Electro-Optical) camera(s). Given a set of stationary ground targets in a terrain (natural, urban, or mixed), the objective is to compute a path for the reconnaissance aircraft so that it can photograph all targets in minimum time. That the targets are situated in terrain plays a significant role because terrain features can occlude visibility. As a result, in order for a target to be photographed, the aircraft must be located where both (1) the target is in close enough range to satisfy the photograph's resolution requirements, and (2) the line-of-sight between the aircraft and the target is not blocked by terrain. For a given target, we call the set of all such aircraft positions the target's *visibility region*. An example visibility region is illustrated in Fig. 1. In full generality, the aircraft path planning can be complicated by wind, airspace constraints (e.g. due to enemy threats or collision avoidance), aircraft dynamic constraints, and the aircraft body itself occluding visibility. However, under simplifying assumptions, if we model the aircraft as a Dubins vehicle^a, approximate the targets' visibility regions by polygons, and let the path be a closed tour, then the reconnaissance path planning problem can be reduced to the following.

For a Dubins vehicle, find a shortest planar closed tour which visits at least one point in each of a set of polygons.

*This version: June 26, 2011

[†]PhD student, Center for Control, Dynamical Systems, and Computation; karl@engr.ucsb.edu. Student Member AIAA.

[‡]PhD student, Department of Mechanical Engineering; paul.v.oberlin@gmail.com

[§]Professor, Department of Mechanical Engineering; dswaroop@tamu.edu

^aA Dubins vehicle is one which moves only forward and has a minimum turning radius.^{3,4}

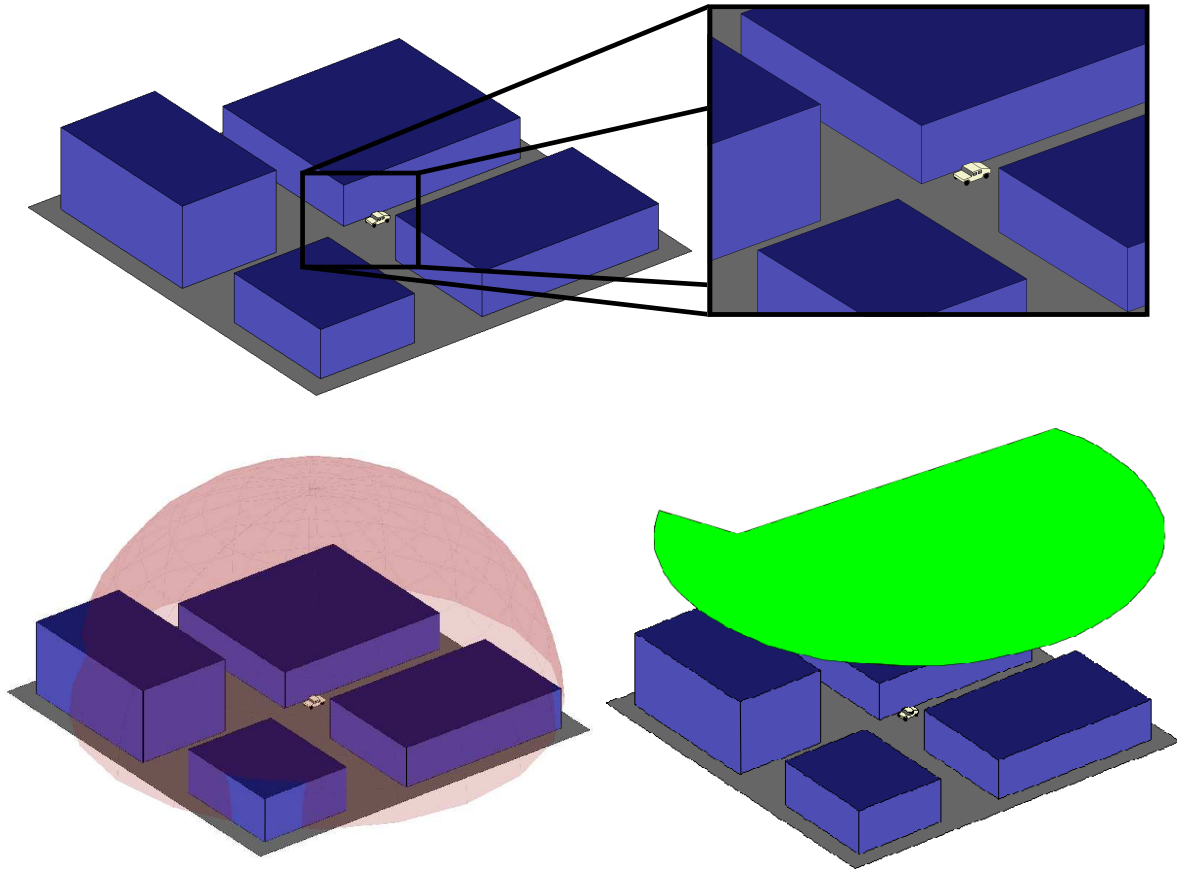


Figure 1. Top shows an example target, a ground vehicle parked next to a building in urban terrain. The set of all points which are close enough to the target to satisfy photograph resolution requirements is a solid sphere (bottom left). The green two-dimensional region in the sky (bottom right) shows the subset of the sphere, at a reconnaissance aircraft's altitude h , where target visibility is not occluded by terrain. Assuming the aircraft body itself doesn't occlude visibility, then flying the aircraft through the green region is sufficient for the target to be photographed, hence we call it the target's *visibility region* for fixed aircraft altitude h .

We refer to this henceforth as the *PVDTSP* (*Polygon-Visiting Dubins Traveling Salesman Problem*) since it is a variation of the famous *TSP* (*Traveling Salesman Problem*).^b A graphical illustration of the PVDTSP is shown in Fig. 2.

I.A. Related Work

To our knowledge the PVDTSP has not previously been studied aside from Ref. 6 where we designed a genetic algorithm. Although the genetic algorithm performs on average fairly well in Monte-Carlo numerical studies, there unfortunately is significant variance in solution quality and no proven performance guarantees. Because the PVDTSP has embedded in it the combinatorial problem of choosing the order to visit the polygons, the solution space is very large and discontinuous. This precludes direct application of numerical optimal control techniques traditionally used in trajectory optimization, surveyed, e.g., in Ref. 7. However, several related variations of the TSP are of interest (summarized in Table 1). The *ETSP* (*Euclidean TSP*) is a TSP where the vertices of the graph are points in the Euclidean plane \mathbb{R}^2 and the edges are weighted with Euclidean distances. In the *ETSPN* (*Euclidean TSP with Neighborhoods*) one seeks a shortest closed Euclidean path passing through n subsets of the plane. The ETSP is NP-hard⁸ and so is the ETSPN by virtue of being a generalization of the ETSP. The *DTSP* (*Dubins TSP*), where a Dubins vehicle must follow a shortest tour through n single point targets in the plane, is known to be NP-hard.¹² Various heuristics for both single and multi-vehicle versions of the DTSP can be found, e.g., in Ref. 13, 14, and 15. The PVDTSP

^bThe TSP, one of the most famous NP-hard problems of combinatorial optimization, is to find a minimum-cost tour (cyclic path) through a weighted graph such that every vertex is visited exactly once. If the graph is directed, it is called the ATSP (Asymmetric TSP). See, e.g., Ref.⁵

Table 1. Relevant Variations of the TSP (Traveling Salesman Problem), all NP-Hard

Name	Brief Description	References
ATSP (Asymmetric TSP)	Find a minimum-cost tour (cyclic path) through a weighted directed graph such that every vertex is visited exactly once	5
STSP (Symmetric TSP)	Find a minimum-cost tour through a weighted undirected graph such that every vertex is visited exactly once	5
ETSP (Euclidean TSP)	Special case of STSP where the vertices of the graph are points in the plane \mathbb{R}^2 and the edges are weighted with Euclidean distances	8
ETSPN (ETSP with Neighborhoods)	Find a minimum-cost Euclidean tour passing through n subsets of the plane	9–11
DTSP (Dubins TSP)	Find a minimum-cost tour for a Dubins vehicle through n single point targets in the plane	12–20
PVDTSP (Polygon-Visiting DTSP)	Find a minimum-cost tour for a Dubins vehicle through n polygons in the plane	6
FOTSP (Finite One-in-set TSP)	Find a minimum-cost tour which passes through at least one vertex in each of a finite collection of <i>clusters</i> , the clusters being mutually exclusive finite vertex sets	5, 21, 22

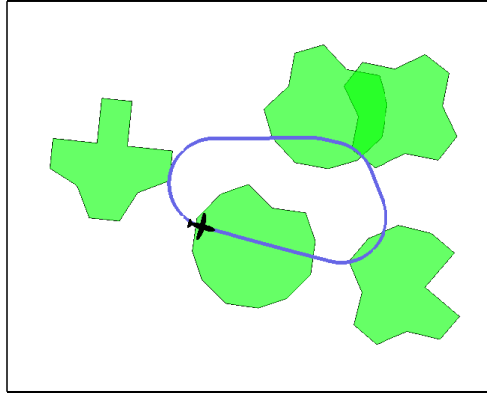


Figure 2. Example problem instance and candidate solution path for the PVDTSP (*Polygon-Visiting Dubins Traveling Salesman Problem*). In order to photograph all targets, the aircraft must fly through at least one point in each target’s visibility region (green), cf. Fig. 1.

reduces to the ETSPN in the limit as the vehicle’s minimum turning radius becomes small compared to the distances between polygons. Similarly, as the area of the polygons goes to zero, the PVDTSP reduces to the DTSP, hence the PVDTSP is NP-hard. There exist a number of algorithms with approximation guarantees for both the DTSP^{16–18} and ETSPN,^{9–11} but it appears that extending any of these algorithms to the PVDTSP would put undesirable restrictions on the problem instances which could be handled, e.g., the polygons would not be allowed to overlap. The *FOTSP (Finite One-in-set TSP)*^c is the problem of finding a minimum-cost closed path which passes through at least one vertex in each of a finite collection of *clusters*, the clusters being mutually exclusive finite vertex sets. The FOTSP is NP-hard because it has as a special case the *ATSP (Asymmetric TSP)*.⁵ An FOTSP instance can be solved exactly by transforming it into an ATSP instance using the *Noon-Bean transformation* from Ref. 21, then invoking an ATSP solver. In the robotics literature,^{23,24} a *sampling-based roadmap method*^d refers to any algorithm which operates by sampling a finite set of points from a continuous state space in order to reduce a continuous motion planning problem to planning on a finite discrete graph called a *roadmap*. Sampling-based roadmap methods have traditionally only been used for collision-free point-to-point path planning amongst obstacles, however, in Ref. 19 approximate solutions to the DTSP are found by sampling discrete sets of orientations that the Dubins vehicle can have over each target, essentially approximating a DTSP instance by an FOTSP instance. The Noon-Bean transformation is then used to convert the FOTSP instance into an ATSP instance so that a standard ATSP solver can be applied. In a reconnaissance context, Ref. 25 solves an FOTSP to decide which of a pair of cameras a UAV should choose to photograph each of a set of targets in sequence. Discretization of the vehicle state space in order to approximate the original problem by an FOTSP is a key idea which we build upon in designing sampling-based roadmap methods for the PVDTSP in the present work.

I.B. Statement of Contribution

There are two main contributions in this article. First, we precisely formulate the general aircraft visual reconnaissance problem for static ground targets in terrain. Under simplifying assumptions, we reduce our general formulation to the PVDTSP. Although the PVDTSP reduces to the well-studied DTSP and ETSP in the *sparse limit* as targets are very far apart and minimum turning radius is small, we provide a worst-case

^cWhat we have chosen to call the FOTSP is known variously in the literature as “Group-TSP”, “Generalized-TSP”, “One-of-a-Set TSP”, “Errand Scheduling Problem”, “Multiple Choice TSP”, “Covering Salesman Problem”, or “International TSP”.

^dIn this usage, “method” means a high level algorithm having multiple components, each of which may be considered an algorithm in its own right.

analysis demonstrating the importance of developing specialized algorithms for the PVDTS in scenarios where targets are close together and polygons may overlap significantly. An early version of the PVDTS formulation appeared in our previous work Ref. 6, but that did not include the worst-case analysis.

Our second main contribution is the design, analysis, and numerical study of two algorithms for the PVDTS. These sampling-based roadmap methods operate by sampling finite discrete sets of poses (positions with orientations) in the target visibility regions in order to approximate a PVDTS instance by an FOTS instance called the *roadmap*. Once a roadmap has been constructed, the algorithms apply the Noon-Bean transformation from Ref. 21 to solve the FOTS. Under certain technical assumptions, the algorithms are *resolution complete*, which means the solution returned converges to a global optimum as the number of samples grows, i.e., as the resolution of the roadmap becomes finer.^e The two algorithms differ only in how they sample poses to construct the roadmap. In the first algorithm, poses are sampled in the interior of the visibility regions. This is a fairly straightforward extension of the angle sampling used for the DTS in Ref. 19. The second algorithm, however, samples so-called *entry poses*. Entry poses are poses which are located on the boundary of a visibility region and are oriented towards the interior of that region. By sampling entry poses, the second algorithm requires slightly stricter assumptions for resolution completeness, but it greatly reduces computation times. While we have borrowed the idea of approximation by an FOTS from Ref. 19, the present work goes beyond a simple extension in that we (1) provide proofs of resolution completeness and (2) use the novel entry-pose sampling technique to reduce computational time complexity. Numerical experiments indicate that our algorithms deliver very good solutions suitably quickly for online purposes when applied to PVDTS instances having up to about 20 targets. They significantly outperformed the alternative approaches of the genetic algorithm in Ref. 6 and DTS over target locations in Ref. 19. Additionally, our algorithms allow a means for a user to trade off computation time for solution quality and their modular nature allows them to easily be extended to handle wind, airspace constraints, any vehicle dynamics, and open-path (vs. closed-tour) problems.

I.C. Organization

This article is organized as follows. In Sec. II we introduce notation, mathematically formulate the minimum-time reconnaissance aircraft path planning problem, show how to reduce the problem to a PVDTS, and provide the worst-case analysis motivating the development of specialized PVDTS algorithms. We present our algorithms in Sec. III and convergence analysis in Sec. IV. We numerically validate the algorithm in Sec. V, describe extensions in Sec. VI, and conclude in Sec. VII.

II. Mathematical Formulation

We begin with some preliminary notation. The s -dimensional Euclidean space is \mathbb{R}^s and \mathbb{S} is the circle parameterized by angle radians ranging from 0 to 2π , 0 and 2π identified. Let $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \dots, \mathcal{T}_n\}$ be the set of n targets which must be photographed by our aircraft. Given a set A , we denote its cardinality by $|A|$, its interior by A° , its closure by \bar{A} , and its power set, i.e., the set of all subsets of A , by 2^A . Given two sets A and B , $A \times B$ is the Cartesian product of these sets. The state of our reconnaissance aircraft is encoded in a vector \mathbf{x} , which takes a value in the aircraft's state space X .

We now define a map $\mathcal{V} : \mathcal{T} \rightarrow 2^X$ from the set of targets to subsets of the aircraft state space. Under this map, $\mathcal{V}(\mathcal{T}_i) \subset X$, called the i th target's *visibility region*, is precisely the set of all aircraft states such that \mathcal{T}_i can be photographed whenever the aircraft is in that state. Let us assume a BVP (Boundary Value Problem) solver is available which calculates the minimum-time aircraft trajectory between any two states \mathbf{x} and \mathbf{x}' , provided a trajectory exists. We treat this minimum time between states as a "black box" distance function denoted by $d(\mathbf{x}, \mathbf{x}')$. Now our **minimum-time reconnaissance path planning problem** can be stated as

$$\begin{aligned} \text{Minimize :} & \quad C(\mathbf{x}_1, \dots, \mathbf{x}_n) = \sum_{i=1}^{n-1} d(\mathbf{x}_i, \mathbf{x}_{i+1}) + d(\mathbf{x}_n, \mathbf{x}_1) \\ \text{Subject To :} & \quad \text{for each } i \in \{1, \dots, n\} \text{ there exists } j \in \{1, \dots, n\} \\ & \quad \text{such that } \mathbf{x}_j \in \mathcal{V}(\mathcal{T}_i), \end{aligned} \tag{1}$$

where the decision variables are the states \mathbf{x}_i ($i = 1, \dots, n$). Once an optimal sequence of states $(\mathbf{x}_1, \dots, \mathbf{x}_n)$ has been chosen, then the minimum-time state-to-state trajectory planner can be used to connect each

^eResolution completeness is a notion of convergence commonly used in the motion planning literature; see, e.g., Ref. 23,24,26 and references therein.

pair of consecutive states, thus we obtain a minimum-time closed reconnaissance tour. Since the complete state space of an aircraft can be very complicated, we simplify the discussion by making the following **main assumptions**.

- (i) The aircraft is modeled as a Dubins vehicle with minimum turning radius r_{\min} , fixed altitude h , and constant airspeed V_a .

Comments: This is common for small low-power UAVs.

- (ii) Regardless of state, the aircraft body never occludes visibility between the camera and a target.

Comments: This holds when either there are multiple cameras covering all angles from the aircraft, or there is a sufficiently flexible gimbaled camera with dynamics faster than the aircraft body dynamics.^f

- (iii) There are no airspace constraints nor wind.

Comments: As we will discuss in Sec. VI, our results can easily be extended to handle wind and no-fly zones.

In accordance with assumption (i), the aircraft dynamics take the form

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} V_a \sin(\psi) \\ V_a \cos(\psi) \\ u \end{bmatrix}, \quad (2)$$

where $(x, y) \in \mathbb{R}^2$ are earth-fixed Cartesian coordinates, $\psi \in \mathbb{S}$ is the azimuth angle, and u is the input to an autopilot system. Assumption (ii) tells us that a target can be photographed independent of aircraft azimuth ψ , therefore we can abstract out the aircraft's internal state so that its state space is reduced to

$$\mathbf{x} = (x, y, \psi) \in X = \mathbb{R}^2 \times \mathbb{S} = \text{SE}(2), \quad (3)$$

and the Visibility regions $\mathcal{V}(\mathcal{T}_1), \dots, \mathcal{V}(\mathcal{T}_n)$ can be represented by their 2-dimensional projections onto \mathbb{R}^2 as shown in Fig. 1 and 2 (though they are subsets of $\text{SE}(2)$). While the visibility regions may contain circular arcs due to the camera range constraint, they can be well approximated by polygons. Hereinafter we refer to the state of a Dubins vehicle interchangeably as “state” or “pose”. The minimum-time path between two Dubins states \mathbf{x} and \mathbf{x}' can be computed very quickly in constant time. Let L denote a left turn motion primitive with radius r_{\min} , R a right turn with radius r_{\min} , and S a straight line segment. The main result of Refs. 3 and 27 is that every optimal Dubins path can be expressed as a sequence which takes one of six possible forms: LRL , RRL , LSL , LSR , RSL , or RSR . One can thus find the optimal path by computing the length of the six possible path forms and selecting the shortest. This provides us with our “black box” distance function $d(\mathbf{x}, \mathbf{x}')$ as it appears in the optimization problem Eq. 1. We write $d_{r_{\min}}(\mathbf{x}, \mathbf{x}')$ hereinafter whenever we wish to emphasize the dependence of the distance function on the vehicle minimum turn radius. We have now reduced our minimum-time reconnaissance path planning problem to a PVDTSP.

In some UAV systems in the field today, target visibility regions are neglected and reconnaissance paths are planned by simply solving the DTSP over the target positions, i.e., the UAV is restricted to pass directly over each target in order to photograph it. The worst-case analysis in the following Theorem II.1 demonstrates, however, that an arbitrarily large relative cost increase can be incurred by solving the DTSP instead of the PVDTSP. This cost increase is most pronounced in the *dense limit* (left in Fig. 3) as targets become very close together, which motivates our development of specialized PVDTSP algorithms for tight urban scenarios especially. In contrast, in the *sparse limit* (right in Fig. 3) when the minimum turning radius and visibility region diameters are much smaller than the distances between targets, there is no significant advantage to solving the PVDTSP over the DTSP nor over the ETSP.

Theorem II.1 (DTSP vs. PVDTSP Worst-Case Analysis). *In a fixed compact subset of the plane \mathbb{R}^2 , solving the DTSP over point targets instead of the PVDTSP over those same targets' visibility regions may incur a cost penalty of order $\Omega(n)$ in the worst case.*^g

Proof. The set of all DTSP tours through n point targets is a subset of all PVDTSP tours through those same targets' visibility regions, therefore the length of a tour that results from solving the PVDTSP to optimality

^fAn omnidirectional camera is another possibility, but they typically have poor resolution.

^gA function $f(n)$ is said to be $\Omega(n)$ if there exist positive constants c and n_0 such that $f(n) \geq cn$ for all $n \geq n_0$.

can be no greater than that of solving the DTSP. Now it suffices to prove the theorem by demonstrating a class of visual reconnaissance problem instances, parameterized by the number of targets n , for which the tour cost when solved as a DTSP is order $\Omega(n)$ (lower bounded) yet only order $O(1)$ (upper bounded) when solved as a PVDTSP. One such class of instances is illustrated left in Fig. 3.^h Given any n noncolinear point targets in the plane, we can linearly scale them until the radius of the circle constructed from any three of them has radius smaller than the Dubins vehicle minimum turn radius r_{\min} . This scaling ensures that, in order to fly a feasible DTSP tour, the aircraft must travel a distance at least πr_{\min} for every two targets. Solving the DTSP over these points would thus cost $\Omega(n)$, yet letting the intersection of the targets visibility regions' contain all the targets, the PVDTSP could be solved with a single minimum turn radius loop and thus cost only $O(1)$. \square

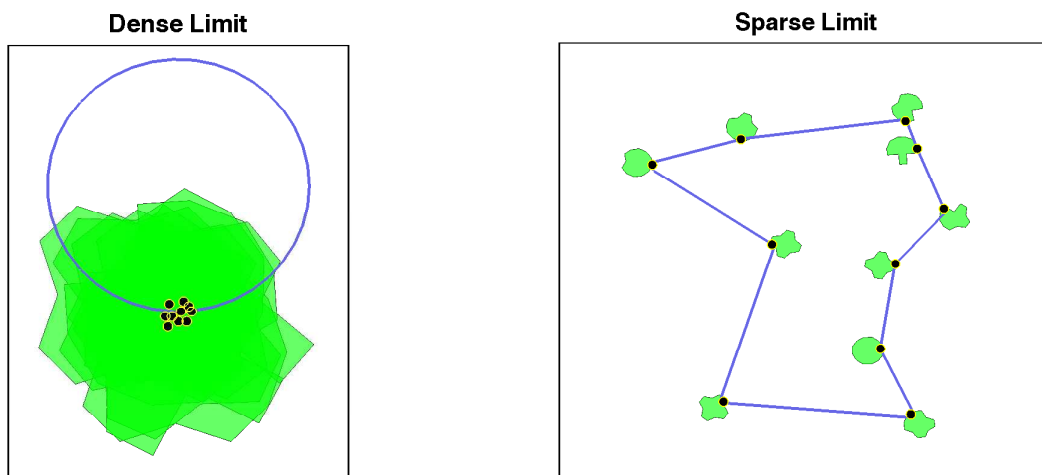


Figure 3. In the *dense limit* (left) as the distances between targets are much smaller than the minimum turning radius, there can be a large penalty incurred ($\Omega(n)$, see Theorem II.1 and proof) by solving the DTSP instead of the PVDTSP. In particular, if the densely packed targets are sufficiently noncolinear, an aircraft solving the PVDTSP can photograph all targets in a single pass (shown as blue circle), but an aircraft solving the DTSP would only be able to photograph two targets per pass, where each pass would be at least πr_{\min} long. In the *sparse limit* (right) when the minimum turning radius and visibility region diameters are much smaller than the distances between targets, there is no significant advantage to solving the PVDTSP over the DTSP nor over the ETSP.

III. Algorithms

We solve PVDTSPs in two main steps which together constitute our sampling-based roadmap methods. In the first step, a finite discrete set of poses is sampled in each polygon in order to approximate a PVDTSP instance by an FOTSP instance. In the second step, the FOTSP is solved by transformation to an ATSP. The approximating FOTSP instance is called the *PVDTSP roadmap* and its structure is made precise through the following definitions.

Definition III.1 (ATSP). Given a weighted directed graph $\mathcal{G} = (V, E)$ where V is a finite set of vertices

$$\{v_1, v_2, v_3, \dots, v_{n_V}\}$$

and E a set of directed edges with weights

$$\{w_{i,j} | i, j \in \{1, 2, 3, \dots, n_V\} \text{ and } i \neq j\},$$

the Asymmetric TSP (ATSP) is to find a directed cycle of minimum cost which visits every vertex in V exactly once.

Definition III.2 (FOTSP). Suppose we have a weighted directed graph $\mathcal{G} = (V, E)$ as in the ATSP Def. III.1, but now the vertices are partitioned into finitely many nonempty mutually exclusive vertex sets called clusters

^hSuch a class of instances has been used previously in Ref. 17 to show DTSP tours have worst-case length $\Omega(n)$.

$S = \{S_1, S_2, S_3, \dots, S_{n_S}\}$. The vertices can thus be written as

$$V = \left\{ v_{(1,1)}, v_{(1,2)}, v_{(1,3)}, \dots, v_{(1,n_{S_1})}, v_{(2,1)}, v_{(2,2)}, v_{(2,3)}, \dots, v_{(2,n_{S_2})}, \dots \right. \\ \left. \dots, v_{(n_S,1)}, v_{(n_S,2)}, v_{(n_S,3)}, \dots, v_{(n_S,n_{S_{n_S}})} \right\}, \quad (4)$$

where $v_{(i,j)}$ is the j th vertex of the cluster S_i . Then the Finite One-in-a-set TSP (FOTSP) is to find a directed cycle of minimum cost which visits at least one vertex from each cluster.

Definition III.3 (PVDTSP Roadmap). A roadmap for a PVDTSP instance is an FOTSP instance, as per Def. III.2, where there is one cluster for each polygon. The vertices V are obtained by sampling a finite set of poses in each polygon and assigning them to their respective clusters. The edges E are obtained by making all possible inter-cluster connections using Dubins minimum-time state-to-state distances as weights.

There are many different possible ways to sample poses for a roadmap. In Sec. III.A we describe one technique based on straightforward extension of the angle sampling for the DTSP in Ref. 19. In Sec. III.B we describe a novel alternative technique which, as we will see in Sec. V, performs significantly better in practice. Once a roadmap has been constructed, we find the best tour in it by the procedure described in Sec. III.C.

III.A. Constructing a Roadmap from Interior Poses

In Ref. 19, a roadmap for the DTSP was constructed by sampling a uniform grid of angles on \mathbb{S} at each target. Each cluster in their roadmap thus consisted of a set of vertices (poses) with the same (x, y) coordinates but different ψ values evenly spaced at intervals of $\delta\psi$ around $[0, 2\pi)$ (see Fig. 4a). A generalization of this technique for the PVDTSP is to simply construct a roadmap by sampling a uniform grid of poses in the interior of each visibility region (Fig. 4b). We refer to a roadmap constructed in this manner as an *interior pose roadmap*.

Table 2 shows a pseudocode for interior pose roadmap construction. The visibility region parameters $\mathcal{V}(\mathcal{T}_1), \dots, \mathcal{V}(\mathcal{T}_n)$ provide the polygons of the PVDTSP instance; r_{\min} is the Dubins vehicle minimum turning radius; \hat{n}_{samples} is an estimate of the number of samples the roadmap should contain; and α is a weighting parameter which determines how many angle samples there will be per (x, y) position in the grid. The larger α is, the more angle samples there will be per (x, y) position. We use an estimate \hat{n}_{samples} instead of the actual number of samples n_{samples} because

- (i) unless the polygons of the problem instance happen to be squares, it is impossible to know a priori how many (x, y) samples on a uniform grid will actually fall into each polygon,
- (ii) in a uniform grid there is a fixed number of ψ samples per (x, y) sample, so not all sample counts are realizable, and
- (iii) it is convenient for a user to only have to keep track of the single input parameter \hat{n}_{samples} rather than three separate grid spacing parameters.

Let δx be the uniform grid spacing in the x direction, δy the spacing in the y direction, and $\delta\psi$ the angular spacing. If the polygons were perfect squares with total area A , $\delta x \delta y$ divided A evenly, and $\delta\psi$ divided 2π evenly, then we would expect that

$$n_{\text{samples}} = \frac{A}{\delta x \delta y} \frac{2\pi}{\delta\psi}. \quad (5)$$

For general nonsquare polygons, the area A on line 2 can be computed, e.g., by triangulating the polygons and summing the areas of the triangles.²⁸ Assuming $\delta x = \delta y = \alpha \delta\psi$ and solving Eq.5 for the spacings gives the formulas on line 3 of Table 2. Using these formulas even when the polygons are not squares, n_{samples} is in practice very close to the user specified \hat{n}_{samples} . Regardless of how close n_{samples} is to \hat{n}_{samples} , a key feature for the convergence proof in Sec. IV is that the grid spacings monotonically go to zero as \hat{n}_{samples} increases.

Once grid spacings have been set, roadmap samples are generated for the polygons separately, one at a time (lines 4-9). For a given polygon represented as a list of vertices, an axis-aligned bounding box B is found simply by selecting the minimum and maximum x and y coordinates over all polygon vertices. A

uniform grid G of poses is then formed on the bounding box according to the spacings δx , δy , and $\delta\psi$. The loop on lines 7-9 is a rejection sampling procedure which ensures that only those grid samples are added to the roadmap which actually lie in the polygon (Fig. 4b, left). Note that even when polygons overlap, their grids are sampled separately. This means that identical pose samples may exist in distinct clusters of the roadmap. If the best PVDTSPTour passes through the intersection of polygons, then the roadmap search procedure to be described in Sec. III.C automatically selects poses in the intersection.

The roadmap construction is completed on lines 10-16 by adding an edge between all pairs of vertices which belong to distinct clusters (Fig. 4b, right). Each edge is weighted by the length $d_{r_{\min}}(\mathbf{x}, \mathbf{x}')$ of the minimum-time Dubins path connecting the respective poses.

Table 2. Interior Pose Roadmap Construction

```

INTERIOR_POSE_ROADMAP(  $\mathcal{V}(\mathcal{T}_1), \dots, \mathcal{V}(\mathcal{T}_n), r_{\min}, \hat{n}_{\text{samples}}, \alpha$  )
  {Initialize Empty Roadmap and Set Grid Spacings}
  1:  $(V, E) \leftarrow (\emptyset, \emptyset)$ ;
  2:  $A \leftarrow$  total of areas of polygons  $\mathcal{V}(\mathcal{T}_1), \dots, \mathcal{V}(\mathcal{T}_n)$ ;
  3:  $\delta x \leftarrow \sqrt[3]{\frac{A\alpha 2\pi}{\hat{n}_{\text{samples}}}}$ ;  $\delta y \leftarrow \delta x$ ;  $\delta\psi \leftarrow \frac{1}{\alpha}\delta x$ ;
  {Sample Clusters}
  4: for  $i = 1$  to  $n$  do
  5:    $B \leftarrow$  axis-aligned bounding box around  $\mathcal{V}(\mathcal{T}_i)$ ;
  6:    $G \leftarrow$  uniform grid on  $B$  with translational spacings  $\delta x$  and  $\delta y$ , and angular spacing  $\delta\psi$ ;
  7:   for all poses  $v$  in  $G$  do
  8:     if  $v \in \mathcal{V}(\mathcal{T}_i)$  then
  9:        $V \leftarrow V \cup v$ ;
  {Connect Clusters}
  10: for  $i = 1$  to  $n$  do
  11:   for  $j = 1$  to  $n$  such that  $j \neq i$  do
  12:     for all vertices  $v$  in cluster  $i$  do
  13:       for all vertices  $v'$  in cluster  $j$  do
  14:          $e \leftarrow$  edge from  $v$  to  $v'$  with weight  $d_{r_{\min}}(v, v')$ ;
  15:          $E \leftarrow E \cup e$ ;
  16: return  $(V, E)$ ;

```

III.B. Constructing a Roadmap from Entry Poses

With very little loss of generality, we can construct a roadmap by sampling poses on the boundaries of the polygons rather than the interiors. Let $\tau^* = (\mathbf{x}_1^*, \dots, \mathbf{x}_n^*)$ be a sequence of poses representing a globally optimal tour for a particular PVDTSPTour instance. Assume we know a priori, even before computing τ^* , that τ^* passes through the boundary of every polygon. Then each pose $\mathbf{x}_1^*, \dots, \mathbf{x}_n^*$ can be taken to be (1) located on the boundary of a polygon and (2) oriented towards the interior of that polygon. We refer to such poses as *entry poses*. To assume τ^* can be represented by a sequence of entry poses is not very restrictive in our experience. If, for example, at least one of the n visibility regions (perhaps belonging to a remote user) is disjoint from the other $n - 1$ visibility regions, then τ^* is guaranteed to pass through the boundary of every visibility region. The space of interior poses we sampled on in Sec. III.A was 3-dimensional, but the space of entry poses, by being restricted to the polygon boundaries, is only 2-dimensional. Intuitively, this difference in dimensionality may allow us to find better tours with fewer samples by building an *entry pose roadmap* (Fig. 4c). Indeed, we will see in the numerical experiments in Sec. V that much faster computation times can be achieved by using entry poses instead of interior poses.

Table 3 shows a pseudocode for entry pose roadmap construction. As for the interior roadmap construction, the parameters are the visibility regions $\mathcal{V}(\mathcal{T}_1), \dots, \mathcal{V}(\mathcal{T}_n)$, vehicle minimum turn radius r_{\min} , sample count estimate \hat{n}_{samples} , and angle density weighting α . For reasons similar to those given in Sec. III.A, we use an estimate \hat{n}_{samples} instead of the actual number of samples n_{samples} . On line 2, the total L of all the perimeter lengths of all the polygons is computed. Let δl be the translational grid spacing along a polygon's boundary and $\delta\psi$ the angular spacing. If δl divided L evenly and $\delta\psi$ divided π evenly, then we would expect

in a uniform grid that

$$n_{\text{samples}} = \frac{L}{\delta l} \frac{\pi}{\delta \psi}. \quad (6)$$

Assuming that $\delta l = \alpha \delta \psi$ and solving Eq.6 for the spacings gives the formulas on line 3 of Table 3. Using these formulas, n_{samples} in practice is very close to the user specified \hat{n}_{samples} . Regardless of how close n_{samples} is to \hat{n}_{samples} , a key feature for the convergence proof in Sec. IV is that the grid spacings monotonically go to zero as \hat{n}_{samples} increases.

Once grid spacings have been set, roadmap samples are generated for the polygons separately, one at a time (lines 4-7). For a given polygon represented as a list of vertices, the uniform grid G of entry poses is formed simply by stepping along the boundary $\partial \mathcal{V}(\mathcal{T}_i)$ at increments of δl and adding $\lfloor \frac{\alpha \pi}{\delta \psi} \rfloor$ samples per step (Fig. 4c, left). The loop on lines 6-7 adds all grid samples to the roadmap. As with the interior pose roadmap construction, even when polygons overlap, their grids are sampled separately. Identical pose samples may thus exist in distinct clusters of the roadmap. If the best PVDTSP tour passes through the intersection of polygons, then the roadmap search procedure to be described in Sec. III.C automatically selects poses in the intersection as necessary.

The roadmap construction is completed on lines 8-13 by adding an edge between all pairs of vertices which belong to distinct clusters (Fig. 4c, right). Each edge is weighted by the length $d_{r_{\min}}(\mathbf{x}, \mathbf{x}')$ of the minimum-time Dubins path connecting the respective poses.

Table 3. Entry-Pose Roadmap Construction

```

ENTRY_POSE_ROADMAP(  $\mathcal{V}(\mathcal{T}_1), \dots, \mathcal{V}(\mathcal{T}_n)$ ,  $r_{\min}$ ,  $\hat{n}_{\text{samples}}$ ,  $\alpha$  )
  {Initialize Empty Roadmap and Set Grid Spacings}
  1:  $(V, E) \leftarrow (\emptyset, \emptyset)$ ;
  2:  $L \leftarrow$  total of perimeter lengths of polygons  $\mathcal{V}(\mathcal{T}_1), \dots, \mathcal{V}(\mathcal{T}_n)$ ;
  3:  $\delta l \leftarrow \sqrt{\frac{L \alpha \pi}{\hat{n}_{\text{samples}}}}$ ;  $\delta \psi \leftarrow \frac{1}{\alpha} \delta l$ ;
  {Sample Clusters}
  4: for  $i = 1$  to  $n$  do
  5:    $G \leftarrow$  uniform grid of entry poses on  $\partial \mathcal{V}(\mathcal{T}_i)$  with translational spacing  $\delta l$  and angular spacing  $\delta \psi$ ;
  6:   for all poses  $v$  in  $G$  do
  7:      $V \leftarrow V \cup v$ ;
  {Connect Clusters}
  8: for  $i = 1$  to  $n$  do
  9:   for  $j = 1$  to  $n$  such that  $j \neq i$  do
  10:    for all vertices  $v$  in cluster  $i$  do
  11:     for all vertices  $v'$  in cluster  $j$  do
  12:       $e \leftarrow$  edge from  $v$  to  $v'$  with weight  $d_{r_{\min}}(v, v')$ ;
  13:       $E \leftarrow E \cup e$ ;
  14: return  $(V, E)$ ;

```

III.C. Finding the Best Tour in a PVDTSP Roadmap

A pseudocode for our roadmap search method is shown in Table 4. This pseudocode represents two different algorithms depending on whether INTERIOR_POSE_ROADMAP or ENTRY_POSE_ROADMAP is substituted for the ROADMAP function on line 1. The inputs are the target visibility regions $\mathcal{V}(\mathcal{T}_1), \dots, \mathcal{V}(\mathcal{T}_n)$, vehicle minimum turn radius r_{\min} , sample count estimate \hat{n}_{samples} , and angle density weighting α . After the roadmap is constructed, the function NOON_BEAN on line 2 transforms the roadmap FOTSP instance (V, E) into an ATSP instance (V', E') by the *Noon-Bean transformation*. The Noon-Bean transformation is defined precisely as follows.

Definition III.4 (Noon-Bean Transformation²¹). *Suppose we are given an FOTSP instance specified, as in Def. III.2, by a weighted directed graph $\mathcal{G} = (V, E)$ together with a partitioning into clusters $S = \{S_1, S_2, S_3, \dots, S_{n_S}\}$. Then the Noon-Bean Transformation of this FOTSP instance is an ATSP instance*

$\mathcal{G}' = (V', E')$ constructed as follows. Begin with $\mathcal{G}' = \mathcal{G}$, i.e., let $V' = V$ and $E' = E$, then make these three modifications to E' :

- (i) For each cluster, add zero-weight directed edges to E' to create a zero-cost cycle which traverses all the vertices of the cluster (so there are a total of n_S zero-cost cycles),
- (ii) cyclically shift intercluster edges of E' so that they emanate from the preceding vertex in their respective zero-cost cycles, and
- (iii) add a large penalty $M = \sum_{i,j} w_{i,j}$, i.e., the total of all weights in \mathcal{G} , to the weight of all intercluster edges in E' .

On line 3, an ATSP solver ATSP_SOLVE is called on the instance (V', E') . Let us assume that we have an ATSP solver at our disposal and the instance (V', E') has been solved. The solution $(v_{k_1}, \dots, v_{k_{n_k}})$ is a list of vertices of the ATSP instance where the sequence $\{k_i\}_{i=1}^{n_k}$ encodes the identifiers of the vertices in the original FOTSP instance. Using these identifiers, the procedure on lines 4-8 extracts the solution $(\mathbf{x}_1, \dots, \mathbf{x}_n)$ to the FOTSP instance. Finally, to produce a sequence of waypoints followable by an aircraft, Dubins minimum-time state-to-state trajectories can be used to interpolate between each pair of states in the solution encoding $(\mathbf{x}_1, \dots, \mathbf{x}_n)$.

As part of our algorithms, we assume access to an ATSP solver. If we want to compute reconnaissance tours suitably quickly for online purposes, then the ATSP solver must be fast. As can be expected for an NP-hard problem, exact ATSP solvers based on branch and bound can take hours to find solutions that an inexact heuristic solver finds in only seconds. Also, state-of-the-art heuristic TSP solvers, e.g., LKH²⁹ or Linkern,³⁰ perform so well in practice that they are widely accepted as *effectively exact* for ATSP instances having up to thousands of vertices. In the implementation of our roadmap method we have therefore chosen to use the powerful LKH software for solving ATSPs. LKH is based on the Lin-Kernighan Heuristic.³¹ To solve an ATSP, it begins with a randomly generated tour. It then attempts to incrementally improve the tour by repeatedly swapping edges, according to so-called *sequential positive-gain lambda-opt moves*, until no further improvement is possible. The inner workings of LKH are very sophisticated, so we refer the interested reader to Ref. 29 for more details.

Table 4. Sampling-Based Roadmap Method for the PVDTS

PVDTS_SOLVE($\mathcal{V}(\mathcal{T}_1), \dots, \mathcal{V}(\mathcal{T}_n), r_{\min}, \hat{n}_{\text{samples}}, \alpha$)
{ Construct Roadmap and Corresponding ATSP Instance }
1: $(V, E) \leftarrow \text{ROADMAP}(\mathcal{V}(\mathcal{T}_1), \dots, \mathcal{V}(\mathcal{T}_n), r_{\min}, \hat{n}_{\text{samples}}, \alpha)$;
2: $(V', E') \leftarrow \text{NOON_BEAN}(V, E)$;
{ Solve ATSP Instance and Extract Best Tour }
3: $(v_{k_1}, \dots, v_{k_{n_k}}) \leftarrow \text{ATSP_SOLVE}(V', E')$;
4: cyclically shift $(v_{k_1}, \dots, v_{k_{n_k}})$ as necessary to ensure v_{k_1} and $v_{k_{n_k}}$ come from distinct clusters;
5: $j \leftarrow 1$;
6: for $i = 1$ to n_k do
7: if v_{k_i} is the first vertex encountered from a particular cluster then
8: $\mathbf{x}_j \leftarrow v_{k_i}$; $j \leftarrow j + 1$;
9: return $(\mathbf{x}_1, \dots, \mathbf{x}_n)$;

IV. Convergence Analysis

Let τ^* be a globally optimal tour for a PVDTS instance. Suppose for a moment that the cost function $C : (\text{SE}(2))^n \rightarrow \mathbb{R}$ in Eq. 1 were continuous. Then provided any roadmap we use contains a tour sufficiently close to τ^* , we would expect PVDTS_SOLVE could find a tour within any given tolerance of $C(\tau^*)$. In other words, continuity of C would imply resolution completeness of our algorithms. Unfortunately, C is not fully continuous because it is a sum of n discontinuous Dubins distances $d(\mathbf{x}, \mathbf{x}')$. However, as we show in the remainder of this section, C does have useful continuity properties which allow us to prove resolution completeness of our algorithms under certain technical assumptions.

IV.A. Continuity Properties and General Position Assumption

Continuity properties of the Dubins distance have been shown in Ref. 32.

Lemma IV.1 (Piecewise Continuity of Dubins Distance³²). *For fixed initial pose, the Dubins distance function $d : \text{SE}(2) \rightarrow \mathbb{R}$ is continuous everywhere except on a set \mathfrak{S}_d of two 2-dimensional smooth surfacesⁱ embedded in the 3-dimensional domain $\text{SE}(2)$. Furthermore, in the limit from one side of each of these discontinuity surfaces, d is continuous up to and including on the surface.^j*

From this we obtain the continuity properties of the PVDTSP cost function.

Lemma IV.2 (Piecewise Continuity of PVDTSP Cost Function). *The PVDTSP cost function $C : (\text{SE}(2))^n \rightarrow \mathbb{R}$ is guaranteed to be continuous everywhere except on a finite set \mathfrak{S}_C of $(3n - 1)$ -dimensional smooth surfaces embedded in its $3n$ -dimensional domain $(\text{SE}(2))^n$. Furthermore, in the limit from one side of each discontinuity surface, C is continuous up to and including on the surface.*

Proof. As a sum of n Dubins distances, C can only be discontinuous at a particular tour $\tau = (\mathbf{x}_1, \dots, \mathbf{x}_n)$ if some pose in the sequence, say \mathbf{x}_i , lies on one of the 2-dimensional discontinuity surfaces \mathfrak{S}_d of the Dubins distance d (applying Lemma IV.2 with respect to initial pose \mathbf{x}_{i-1}). Let us call the 2-dimensional surface where this occurs \mathfrak{s} . Then $(\text{SE}(2))^{i-1} \times \mathfrak{s} \times (\text{SE}(2))^{n-i}$ is one of the $(3n - 1)$ -dimensional surfaces in \mathfrak{S}_C . Taking the union of all surfaces constructed in this manner for $i = 1, \dots, n$, we obtain all of \mathfrak{S}_C . There thus are only $2n$ surfaces in \mathfrak{S}_C and each inherits the smoothness and one-sided continuity from \mathfrak{S}_d . \square

These continuity properties motivate a *general position assumption* that will allow us to prove resolution completeness of our algorithms. General position assumptions, also known as *generic input assumptions*, are commonly used in the field of computational geometry for proving correctness of algorithms in the absence of certain input degeneracies which almost never occur.³³ We define a PVDTSP instance as being in *general position* if it is not degenerate according to either of the following definitions.

Definition IV.3 (Interior Pose Degeneracy). *A PVDTSP instance is interior pose degenerate if there exists a tour $\tau^* = (\mathbf{x}_1^*, \dots, \mathbf{x}_n^*) \in \mathcal{V}(\mathcal{T}_1) \times \dots \times \mathcal{V}(\mathcal{T}_n)$ and $\delta > 0$ such that for every open set $A \subset \mathcal{V}(\mathcal{T}_1) \times \dots \times \mathcal{V}(\mathcal{T}_n)$ it holds that $\sup_{\tau \in A} C(\tau) \geq C(\tau^*) + \delta$.*

Definition IV.4 (Entry Pose Degeneracy). *Let $\check{\mathcal{V}}(\mathcal{T}_i)$ be the set of entry poses of $\mathcal{V}(\mathcal{T}_i)$ for $i = 1, \dots, n$. A PVDTSP instance is entry pose degenerate if there exists a tour $\tau^* = (\mathbf{x}_1^*, \dots, \mathbf{x}_n^*) \in \partial\mathcal{V}(\mathcal{T}_1) \times \dots \times \partial\mathcal{V}(\mathcal{T}_n)$ and $\delta > 0$ such that for every open^k set $A \subset \check{\mathcal{V}}(\mathcal{T}_1) \times \dots \times \check{\mathcal{V}}(\mathcal{T}_n)$ it holds that $\sup_{\tau \in A} C(\tau) \geq C(\tau^*) + \delta$.*

In words, a PVDTSP instance is degenerate if there exists a tour τ^* such that no matter how finely we may sample interior (resp. entry) poses, we cannot hope that any tour constructed from the sampling comes within a prescribed tolerance δ of the cost $C(\tau^*)$. An example of a PVDTSP instance which is both interior pose and entry pose degenerate is shown in Fig. 6. The globally optimal tour τ^* is a minimum turn radius circle which just grazes the outside corners of the triangular visibility regions. Any tour $\tau \neq \tau^*$ would have to veer away from the visibility regions and then come back in order to visit them all. The cost $C(\tau)$ is thus bounded away from $C(\tau^*)$.

As long as the input data of a PVDTSP instance is subject to some noise, then degeneracies should almost never occur, i.e., a degenerate problem instance will occur with probability 0. A completely rigorous proof of this would require a highly technical digression into intersection theory.³⁴ We therefore content ourselves with the general position assumption based on geometric intuition from the following lower-dimensional optimization problem. Suppose we want to minimize a function $f(x, y)$ over a unit square in \mathbb{R}^2 , where

$$f(x, y) = \begin{cases} 0 & \text{for } \sqrt{x^2 + y^2} \leq 1 \\ 1 & \text{else} \end{cases} \quad (7)$$

ⁱThese discontinuity surfaces of the Dubins distance function are traced out by two circular arcs in the (x, y) plane which vary continuously along the ψ axis. Illustrations are provided in Ref. 32.

^jNote that, for our purposes, it is not important which side of the discontinuity surface is continuous up to and including on the surface.

^kBy an “open set in $A \in \check{\mathcal{V}}(\mathcal{T}_1) \times \dots \times \check{\mathcal{V}}(\mathcal{T}_n)$ ”, we intend “open” in the relative topology of $\check{\mathcal{V}}(\mathcal{T}_1) \times \dots \times \check{\mathcal{V}}(\mathcal{T}_n)$ as a subspace of $(\text{SE}(2))^n$.

The optimal solution clearly depends on how the square is situated relative to the unit disk (see Fig. 7). One could easily solve this problem in closed form by checking the disk and square for intersection. However, in order to make an analogy with our PVDTSP algorithms, suppose our strategy for minimizing f is to sample a uniform grid of points in the interior of the square and then select the sample with minimum f value. Notice that f , similar to C , is piecewise continuous and, in the limit from one side of the (unit circle) discontinuity surface, it is continuous up to and including on the surface. These continuity properties imply that in the limit as the grid becomes finer in the square, the value returned by the sampling strategy should always approach the optimum as long as the disk and square share an open set whenever they intersect. If the disk and square do intersect but there is no such open set, we say the problem instance is *degenerate*. Degeneracy occurs precisely when the disk only intersects the square at a single point, i.e., when the disk is tangent to the square or it touches only at a corner. Given any degenerate problem instance, perturbing that instance by a normal random variable in \mathbb{R}^2 would result in a new problem instance which is degenerate with probability zero. In this sense, degenerate problem instances almost never occur. Completing our analogy with the PVDTSP: C is to f as \mathfrak{S}_C is to the unit circle as visibility regions are to the square. We thus expect, as long as the interaction between the discontinuity surfaces \mathfrak{S}_C and the visibility regions is well-behaved, that a PVDTSP instance will be in general position and our algorithms will converge.

IV.B. Resolution Completeness

We are now ready to state the main convergence results.

Theorem IV.5 (Interior Pose Roadmap Convergence). *Let $\mathcal{V}(\mathcal{T}_1), \dots, \mathcal{V}(\mathcal{T}_n)$ be visibility regions constituting a PVDTSP instance in general position. Let $\{\mathcal{R}_i\}_{i=1}^\infty$ be a sequence of roadmaps constructed according to Def. III.3 such that the vertices of \mathcal{R}_i are dense in the visibility regions $\mathcal{V}(\mathcal{T}_1), \dots, \mathcal{V}(\mathcal{T}_n)$ in the limit as i goes to infinity. Let $\{\tau_i\}_{i=1}^\infty$ be the sequence of best tours contained in the sequence of roadmaps $\{\mathcal{R}_i\}_{i=1}^\infty$, respectively. Then the best tour cost $C(\tau_i)$ approaches a global optimum, i.e.,*

$$\lim_{i \rightarrow \infty} C(\tau_i) = \inf_{\tau \in \mathcal{V}(\mathcal{T}_1) \times \dots \times \mathcal{V}(\mathcal{T}_n)} C(\tau). \quad (8)$$

Proof. To prove Eq. 8 it suffices to show that for all $\epsilon > 0$ there exists N such that $i > N$ implies

$$C(\tau_i) \leq \inf_{\tau \in \mathcal{V}(\mathcal{T}_1) \times \dots \times \mathcal{V}(\mathcal{T}_n)} C(\tau) + \epsilon. \quad (9)$$

By definition of infimum, there exists a sequence of tours $\{\tau'_j\}_{j=1}^\infty$ in $\mathcal{V}(\mathcal{T}_1) \times \dots \times \mathcal{V}(\mathcal{T}_n)$ such that

$$\lim_{j \rightarrow \infty} C(\tau'_j) = \inf_{\tau \in \mathcal{V}(\mathcal{T}_1) \times \dots \times \mathcal{V}(\mathcal{T}_n)} C(\tau),$$

i.e., for all $\epsilon > 0$ there exists N_1 such that $j > N_1$ implies

$$C(\tau'_j) \leq \inf_{\tau \in \mathcal{V}(\mathcal{T}_1) \times \dots \times \mathcal{V}(\mathcal{T}_n)} C(\tau) + \frac{\epsilon}{2}. \quad (10)$$

The general position assumption (specifically the absence of degeneracy per Def. IV.3) implies that for each τ'_j and $\epsilon > 0$ there exists an open set $A_j \subset \mathcal{V}(\mathcal{T}_1) \times \dots \times \mathcal{V}(\mathcal{T}_n)$ such that $\sup_{\tau' \in A_j} C(\tau') \leq C(\tau'_j) + \frac{\epsilon}{2}$. Because the roadmap poses are sampled densely in the limit, we can always find an N_2 large enough that each \mathcal{R}_i contains a tour τ_i in the open set A_j for all $i > N_2$, hence

$$C(\tau_i) \leq C(\tau'_j) + \frac{\epsilon}{2} \quad (11)$$

whenever $i > N_2$. Combining Eq. 10 and 11, we obtain the desired result that for all $\epsilon > 0$ there exists $N = \max\{N_1, N_2\}$ such that $i > N$ implies

$$\begin{aligned} C(\tau_i) &\leq C(\tau'_j) + \frac{\epsilon}{2} \\ &\leq \inf_{\tau \in \mathcal{V}(\mathcal{T}_1) \times \dots \times \mathcal{V}(\mathcal{T}_n)} C(\tau) + \frac{\epsilon}{2} + \frac{\epsilon}{2} \\ &= \inf_{\tau \in \mathcal{V}(\mathcal{T}_1) \times \dots \times \mathcal{V}(\mathcal{T}_n)} C(\tau) + \epsilon. \end{aligned}$$

□

Theorem IV.6 (Entry Pose Roadmap Convergence). *Let $\mathcal{V}(\mathcal{T}_1), \dots, \mathcal{V}(\mathcal{T}_n)$ be visibility regions constituting a PVDTSPP instance in general position. Let $\{\mathcal{R}_i\}_{i=1}^\infty$ be a sequence of roadmaps constructed according to Def. III.3 such that the vertices of \mathcal{R}_i are dense in the entry pose sets $\check{\mathcal{V}}(\mathcal{T}_1), \dots, \check{\mathcal{V}}(\mathcal{T}_n)$ in the limit as i goes to infinity. Let $\{\tau_i\}_{i=1}^\infty$ be the sequence of best tours contained in the sequence of roadmaps $\{\mathcal{R}_i\}_{i=1}^\infty$, respectively. Then the best tour cost $C(\tau_i)$ approaches a global optimum among all tours which pass through the boundary of every visibility region, i.e.,*

$$\lim_{i \rightarrow \infty} C(\tau_i) = \inf_{\tau \in \partial\mathcal{V}(\mathcal{T}_1) \times \dots \times \partial\mathcal{V}(\mathcal{T}_n)} C(\tau). \quad (12)$$

Proof. To prove Eq. 12 it suffices to show that for all $\epsilon > 0$ there exists N such that $i > N$ implies

$$C(\tau_i) \leq \inf_{\tau \in \partial\mathcal{V}(\mathcal{T}_1) \times \dots \times \partial\mathcal{V}(\mathcal{T}_n)} C(\tau) + \epsilon. \quad (13)$$

By definition of infimum, there exists a sequence of tours $\{\tau'_j\}_{j=1}^\infty$ in $\partial\mathcal{V}(\mathcal{T}_1) \times \dots \times \partial\mathcal{V}(\mathcal{T}_n)$ such that

$$\lim_{j \rightarrow \infty} C(\tau'_j) = \inf_{\tau \in \partial\mathcal{V}(\mathcal{T}_1) \times \dots \times \partial\mathcal{V}(\mathcal{T}_n)} C(\tau),$$

i.e., for all $\epsilon > 0$ there exists N_1 such that $j > N_1$ implies

$$C(\tau'_j) \leq \inf_{\tau \in \partial\mathcal{V}(\mathcal{T}_1) \times \dots \times \partial\mathcal{V}(\mathcal{T}_n)} C(\tau) + \frac{\epsilon}{2}. \quad (14)$$

The general position assumption (specifically the absence of degeneracy per Def. IV.4) implies that for each τ'_j and $\epsilon > 0$ there exists an open set $A_j \subset \check{\mathcal{V}}(\mathcal{T}_1) \times \dots \times \check{\mathcal{V}}(\mathcal{T}_n)$ such that $\sup_{\tau' \in A_j} C(\tau') \leq C(\tau'_j) + \frac{\epsilon}{2}$. Because the roadmap poses are sampled densely in the limit, we can always find an N_2 large enough that each \mathcal{R}_i contains a tour τ_i in the open set A_j for all $i > N_2$, hence

$$C(\tau_i) \leq C(\tau'_j) + \frac{\epsilon}{2} \quad (15)$$

whenever $i > N_2$. Combining Eq. 14 and 15, we obtain the desired result that for all $\epsilon > 0$ there exists $N = \max\{N_1, N_2\}$ such that $i > N$ implies

$$\begin{aligned} C(\tau_i) &\leq C(\tau'_j) + \frac{\epsilon}{2} \\ &\leq \inf_{\tau \in \partial\mathcal{V}(\mathcal{T}_1) \times \dots \times \partial\mathcal{V}(\mathcal{T}_n)} C(\tau) + \frac{\epsilon}{2} + \frac{\epsilon}{2} \\ &= \inf_{\tau \in \partial\mathcal{V}(\mathcal{T}_1) \times \dots \times \partial\mathcal{V}(\mathcal{T}_n)} C(\tau) + \epsilon. \end{aligned}$$

□

The resolution completeness of our algorithms now follows directly from the roadmap convergence theorems.

Corollary IV.7 (Resolution Completeness with Interior Pose Sampling). *For a PVDTSPP instance in general position, suppose the algorithm PVDTSPP-SOLVE in Table 4 is executed for $\hat{n}_{\text{samples}} = 1, 2, 3, \dots$, each time returning a tour $\tau_{\hat{n}_{\text{samples}}}$. Suppose further that*

- (i) *for the ROADMAP function on line 1, a roadmap construction as per Def. III.3 is used which samples poses densely in the visibility regions $\mathcal{V}(\mathcal{T}_1), \dots, \mathcal{V}(\mathcal{T}_n)$ in the limit as \hat{n}_{samples} goes to infinity, e.g., the INTERIOR_POSE_ROADMAP function in Table 2, and*
- (ii) *the function ATSP-SOLVE on line 3 always returns an exact solution.*

Then the best tour cost $C(\tau_{\hat{n}_{\text{samples}}})$ approaches a global optimum, i.e.,

$$\lim_{\hat{n}_{\text{samples}} \rightarrow \infty} C(\tau_{\hat{n}_{\text{samples}}}) = \inf_{\tau \in \mathcal{V}(\mathcal{T}_1) \times \dots \times \mathcal{V}(\mathcal{T}_n)} C(\tau). \quad (16)$$

Proof. It is proven in Ref. 21 that the Noon-Bean transformation is exact in the sense that if the exact solution of the ATSP instance is found, then the extracted solution to the FOTSPP instance will also be exact. This means the method will always find the best tour in a given roadmap. The corollary thus follows directly from the roadmap convergence Theorem IV.5. □

Corollary IV.8 (Resolution Completeness with Entry Pose Sampling). *For a PVDTSP instance in general position, suppose the algorithm PVDTSP_SOLVE in Table 4 is executed for $\hat{n}_{\text{samples}} = 1, 2, 3, \dots$, each time returning a tour $\tau_{\hat{n}_{\text{samples}}}$. Suppose further that*

- (i) *for the ROADMAP function on line 1, a roadmap construction as per Def. III.3 is used which samples poses densely in the entry pose sets $\check{\mathcal{V}}(\mathcal{T}_1), \dots, \check{\mathcal{V}}(\mathcal{T}_n)$ in the limit as \hat{n}_{samples} goes to infinity, e.g., the ENTRY_POSE_ROADMAP function in Table 2, and*
- (ii) *the function ATSP_SOLVE on line 3 always returns an exact solution.*

Then the best tour cost $C(\tau_{\hat{n}_{\text{samples}}})$ approaches a global optimum among all tours which pass through the boundary of every visibility region, i.e.,

$$\lim_{\hat{n}_{\text{samples}} \rightarrow \infty} C(\tau_{\hat{n}_{\text{samples}}}) = \inf_{\tau \in \partial\mathcal{V}(\mathcal{T}_1) \times \dots \times \partial\mathcal{V}(\mathcal{T}_n)} C(\tau). \quad (17)$$

Proof. Follows directly from the roadmap convergence Theorem IV.6 as Corollary IV.7 followed from Theorem IV.5. \square

Corollaries IV.7 and IV.8 are valid for any roadmap constructions which sample densely on the visibility sets, resp. entry pose sets. Instead of uniform grids, one could sample, e.g., randomly or quasirandomly.^{23,24,35}

IV.C. Time Complexity

The convergence results in Sec. IV.B show that the tour returned by our algorithm PVDTSP_SOLVE converges to a global optimum, but they tell us nothing about the rate of convergence. In particular, we have no way of knowing how many samples are necessary in order to guarantee the tour returned comes within a prescribed tolerance of the global optimum. In this respect, the utility of the convergence theorems is more conceptual than practical and it remains an important open problem to prove the rate of convergence. In fact, convergence rates are not known for a number of state-of-the-art sampling-based roadmap methods in the literature, yet these methods perform compellingly well in practice.^{24,36} That the PVDTSP is NP-hard also indicates that there can be no polynomial time complexity guarantees. Despite all this, we are able to provide an estimate of time complexity as a function of the input parameter \hat{n}_{samples} , and numerical experiments in Sec. V show the algorithms perform very well in practice. We derive the estimate by assuming an ATSP instance with n_V vertices can be solved in time $\mathcal{O}(n_V^{2.2})$. This is the empirically determined average-case time complexity of state-of-the-art heuristic ATSP solvers based on the Lin-Kernighan heuristic.^{29,31,37}

Proposition IV.9 (Time Complexity). *Suppose whenever we execute the algorithm PVDTSP_SOLVE in Table 4 it holds that*

- (i) $n \ll \hat{n}_{\text{samples}}$,
- (ii) $\hat{n}_{\text{samples}} \approx n_{\text{samples}}$,
- (iii) *the number of vertices representing each polygon is upper bounded¹,*
- (iv) *the roadmap is constructed using INTERIOR_POSE_ROADMAP in Table 2 or ENTRY_POSE_ROADMAP in Table 3, and*
- (v) *an ATSP solver based on the Lin-Kernighan heuristic, e.g., LKH²⁹ or Linkern.³⁰ is able to solve ATSP instances with n_V vertices in time $\mathcal{O}(n_V^{2.2})$.*

Then the time complexity of PVDTSP_SOLVE is

$$\mathcal{O}(\hat{n}_{\text{samples}}^{2.2}). \quad (18)$$

¹In the examples we considered, the number of vertices representing any polygon never exceeded 20.

Proof. We first examine the INTERIOR_POSE_ROADMAP pseudocode Table 2. Computing the area and bounding box of a single polygon takes time linear in its number of vertices,²⁸ which we assumed constant. These computations are performed for n polygons, so lines 1-6 take time $\mathcal{O}(n)$. Lines 7-9 makes roughly $\mathcal{O}(n_{\text{samples}})$ constant-time insertions into G and therefore takes $\mathcal{O}(n_{\text{samples}})$ time. We know these insertions take roughly constant time for us because (1) checking inclusion in a polygon²⁸ is linear in its number of its vertices, and (2) the fraction of rejected samples is proportional to the ratio of polygon area to bounding box area, which is fixed. On lines 10-15, the $\mathcal{O}(n_{\text{samples}}^2)$ edges of a nearly complete graph are inserted, each in constant time. This last term dominates, hence we expect the time complexity of INTERIOR_POSE_ROADMAP to be $\mathcal{O}(n_{\text{samples}}^2)$.

We turn our attention to the ENTRY_POSE_ROADMAP pseudocode Table 3. Computing the perimeter length of n polygons takes $\mathcal{O}(n)$ time (line 2). The cluster sampling on lines 4-7 makes no rejections, so $\mathcal{O}(n_{\text{samples}})$ constant-time insertions take $\mathcal{O}(n_{\text{samples}})$ time. On lines 8-13, the $\mathcal{O}(n_{\text{samples}}^2)$ edges of a nearly complete graph are inserted, each in constant time. This last term dominates, hence we expect the time complexity of ENTRY_POSE_ROADMAP to be $\mathcal{O}(n_{\text{samples}}^2)$.

Finally, we examine the PVDTSP_SOLVE pseudocode Table 4 itself. As we have just seen, building the roadmap (line 1) takes time $\mathcal{O}(n_{\text{samples}}^2)$. The Noon-Bean transformation (line 2) also takes time $\mathcal{O}(n_{\text{samples}}^2)$ because it adds only n_{samples} zero-weight edges, but modifies one at a time the other $\mathcal{O}(n_{\text{samples}}^2)$ edges. Under assumption (v), the ATSP on line 3 can be solved in time $\mathcal{O}(n_{\text{samples}}^{2.2})$. Extracting the PVDTSP solution (lines 4-8) from the ATSP solution takes only $\mathcal{O}(n_{\text{samples}})$ time.

Of all the operations in PVDTSP_SOLVE, solving the ATSP dominates, hence the time complexity is $\mathcal{O}(n_{\text{samples}}^{2.2})$, or $\mathcal{O}(\hat{n}_{\text{samples}}^{2.2})$ assuming $\hat{n}_{\text{samples}} \approx n_{\text{samples}}$. \square

V. Numerical Study

We have implemented the algorithms of Sec. III in C++ on a 2.33 GHz i686. For solving ATSP instances, our implementations call the powerful LKH²⁹ solver as a subroutine. Out of several dozen problem instances we experimented with, the results from three representative examples are shown in Table 5 and Fig. 8, 9, and 10. In all examples the aircraft minimum turn radius was $r_{\min} = 3$ m. We tested many different values of the parameter α ranging from 0 to 6 and found that $\alpha = 2.2$ was best when using INTERIOR_POSE_ROADMAP and $\alpha = 2.85$ was best when using ENTRY_POSE_ROADMAP. While no α values are optimal for all problem instances, these “best” values were found by averaging those values which resulted in fastest convergence over a dozen typical-case problem instances. Using these best parameter values, we ran each algorithm over a range of sample counts for each instance. As predicted by Corollaries IV.7 and IV.8, the algorithms appear to converge in the plots of tour cost vs. number of samples.^m All algorithms perform suitably quickly for online purposes and the plots of computation time vs. sample count appear to be roughly quadratic, which matches the predicted time complexity of Proposition IV.9. The plots also demonstrate that a user can indirectly trade off computation time for solution quality by adjusting the number of samples.

To compare the algorithms, we considered each to have converged when the cost of the returned tour changed less than 5% for 3 successive runs. Table 5 shows these converged values. PVDTSP solutions took more time to compute than the DTSP solutions, but the PVDTSP solutions converged to substantially smaller values. In fact, in the three examples in Table 5, the cost of the DTSP solutions are all at least 35% greater than the cost of the respective PVDTSP solutions. Among the PVDTSP solutions, those found using INTERIOR_POSE_ROADMAP were comparable in cost to those found using ENTRY_POSE_ROADMAP, but the latter required significantly less computation time. In all examples we experimented with, using PVDTSP_SOLVE with ENTRY_POSE_ROADMAP instead of INTERIOR_POSE_ROADMAP consistently reduced computation time by 50% or more. The problem instances we experimented with are the same as those used for testing the genetic algorithm in Ref. 6. The genetic algorithm performed comparable to PVDTSP_SOLVE (with either INTERIOR_POSE_ROADMAP or ENTRY_POSE_ROADMAP) for around 5 targets, slightly worse for 10 targets, and much worse for 20+ targets. In conclusion, PVDTSP_SOLVE with ENTRY_POSE_ROADMAP is the best known algorithm for solving a PVDTSP instance where the solution is guaranteed to pass through the boundary of every polygon. If the solution is not guaranteed to

^mDespite the convergence, the plots of tour cost vs. sample count also show some slight nonmonotonicities. We attribute these monotonicities to the fact that a uniform grid roadmap cannot contain another unless the former has more nodes by a power of two. In other words, a roadmap with low resolution may by chance contain a very good solution while another roadmap with higher resolution may not contain that same solution unless the resolution is double or more.

pass through the boundary of every polygon, then PVDTSP_SOLVE with INTERIOR_POSE_ROADMAP is best.

Table 5. Statistics from example solutions computed in C++ on a 2.33 GHz i686. Each algorithm was run over a range of sample counts (cf. Fig. 8, 9, and 10). Each row in this table represents the first run for which the cost of the returned tour changed less than 5% over the previous 3 consecutive runs.

Example	Algorithm	α	\hat{n}_{samples}	n_{samples}	Computation Time	Tour Cost
Fig. 8 5 targets	DTSP Solver from Ref. 19	N/A	120	120	0.30 s	55.9 m
	PVDTSP_SOLVE with INTERIOR_POSE_ROADMAP	2.2	450	491	7.53 s	41.4 m
	PVDTSP_SOLVE with ENTRY_POSE_ROADMAP	2.85	350	350	3.50 s	37.5 m
Fig. 9 10 targets	DTSP Solver from Ref. 19	N/A	250	250	1.71 s	104.3 m
	PVDTSP_SOLVE with INTERIOR_POSE_ROADMAP	2.2	550	623	16.27 s	73.8 m
	PVDTSP_SOLVE with ENTRY_POSE_ROADMAP	2.85	450	468	7.97 s	69.9 m
Fig. 10 20 targets	DTSP Solver from Ref. 19	N/A	350	360	4.72 s	194.3 m
	PVDTSP_SOLVE with INTERIOR_POSE_ROADMAP	2.2	1000	1113	99.98 s	124.6 m
	PVDTSP_SOLVE with ENTRY_POSE_ROADMAP	2.85	550	708	29.80 s	128.4 m

VI. Extensibility

VI.A. Handling Wind, Airspace Constraints, and Any Vehicle Dynamics

In Sec. II we formulated the minimum-time reconnaissance path planning problem as finding a sequence of states $(\mathbf{x}_1, \dots, \mathbf{x}_n)$ from which the targets can be photographed. We assumed a minimum-time state-to-state trajectory planner was available as a “black box” that could be accessed by our algorithms in order to evaluate the distance function $d(\mathbf{x}, \mathbf{x}')$, which is all we need to evaluate the goodness of any candidate solution. In this way, the minimum-time state-to-state trajectory planner is a *module* within our algorithms. We could therefore use our algorithms with any of the minimum-time state-to-state trajectory planners available in the literature. These include planners which can handle wind, no-fly zones, and any vehicle dynamics. The literature on nonholonomic trajectory planning is vast, so we survey only briefly a few works most relevant.

Without obstacles or wind, the procedure for computing an optimal Dubins pose-to-pose path was shown using geometric arguments in Ref. 3, then later more concisely using Pontryagin’s maximum principle from optimal control in Ref. 27.³⁸ More recently it has been shown that in a constant wind field without obstacles, a shortest pose-to-pose Dubins path can be calculated in constant time to fixed precision.^{4, 39, 40} Using these methods, pose-to-pose shortest path queries with no obstacles can be computed in constant time.

Given a polygonal environment with polygonal holes represented by a total of m vertices, the shortest collision-free Euclidean path (no curvature constraint) can be calculated in $\mathcal{O}(m \log m)$ time.⁴¹ Unfortunately the same problem with a Dubins vehicle is NP-hard in m .⁴² However, much work has been done to quickly find nearly optimal obstacle avoiding paths, surveyed further in Ref. 23, 24, 26. Trajectory plan-

³⁸It is a lesser known fact that the problem of finding bounded curvature paths was first studied by Markov in connection with piecing together railroad tracks. An account of his results can be found in Ref. 38.

ners specifically intended for fixed-wing UAVs, which use a branch and bound technique, are described in Ref. 43, 44. Another approach to nonholonomic motion planning with obstacles is to use a MILP (Mixed Integer Linear Program).^{45, 46}

VI.B. Open-Path vs. Closed-Tour Problems

In presenting our algorithms, we considered only closed-tour solutions to the reconnaissance UAV path planning problem. One may alternatively wish to find an open reconnaissance path from a fixed initial pose $\mathbf{x}_{\text{initial}}$ to a different fixed final pose $\mathbf{x}_{\text{final}}$. In this case, instead of the closed-tour cost function in Eq. 1, we use the *open path cost function*

$$C(\mathbf{x}_1, \dots, \mathbf{x}_n) = d(\mathbf{x}_{\text{initial}}, \mathbf{x}_1) + \sum_{i=1}^{n-1} d(\mathbf{x}_i, \mathbf{x}_{i+1}) + d(\mathbf{x}_n, \mathbf{x}_{\text{final}}).$$

The algorithms can be applied to open-path problems with only slight modification in how the roadmap is constructed. The open-path roadmap has all the vertices and edges that the closed-tour roadmap of Def. III.3 does, but in addition has two single-vertex clusters, one for the initial pose and one for the final pose. The initial single-vertex cluster is connected by distance d -weighted edges outgoing to all vertices in nondegenerate clusters. The final single-vertex cluster is connected (1) by a zero-weight edge outgoing to the initial cluster vertex, and (2) by distance d -weighted edges incoming from all vertices in the nondegenerate clusters.

VII. Conclusion

We have formulated the general aircraft visual reconnaissance problem for static ground targets in terrain and shown that it can be reduced to a new variant of the Traveling Salesman Problem called the PVDTS. A worst-case analysis demonstrated the importance of developing specialized algorithms for the PVDTS in scenarios where targets are close together and polygons may overlap significantly. We designed two algorithms for the PVDTS. These sampling-based roadmap methods operate by sampling finite discrete sets of poses in the target visibility regions in order to approximate a PVDTS instance by an FOTS instance called the roadmap. Once a roadmap has been constructed, the algorithms apply the Noon-Bean transformation to solve the FOTS. Under certain technical assumptions, the algorithms are resolution complete. The two algorithms differ only in how they sample poses to construct the roadmap. In the first algorithm, poses are simply sampled in the interior of the visibility regions. The second algorithm, however, samples entry poses. Sampling entry poses requires slightly stricter assumptions for resolution completeness, but greatly reduces computation time. Numerical experiments indicate that our algorithms deliver very good solutions suitably quickly for online purposes when applied to PVDTS instances having up to about 20 targets. They significantly outperformed the alternative approaches of the genetic algorithm in Ref. 6 and DTSP over target locations in Ref. 19. Additionally, our algorithms allow a means for a user to trade off computation time for solution quality and their modular nature allows them to easily be extended to handle wind, airspace constraints, any vehicle dynamics, and open-path (vs. closed-tour) problems.

While the algorithms we have presented are essentially ready to be fielded, there is much room for future work. We are currently investigating extensions to multiple vehicles, constant factor approximation guarantees, and a way to calculate how many samples a roadmap needs to guarantee a prescribed accuracy. Aside from improvements to the existing algorithms, it would be interesting to numerically evaluate hybrid approaches.

Acknowledgments

This work was supported by a US Department of Defense SMART Graduate Fellowship (<http://www.asee.org/fellowships/smart/>). Thanks to the following people for helpful comments: F. Bullo (UCSB), R. Holsapple (WPAFB), D. Kingston (WPAFB), M. Mears (WPAFB), F. Obermeyer (Toyon Research Corp.), S. Rasmussen (Miami Valley Aerospace LLC), C. Schumacher (WPAFB), V. Shaferman (Technion), R. Spjut (UCSB).

References

- ¹Chandler, P., Pachter, M., and Rasmussen, S., "UAV Cooperative Control," *American Control Conference*, Arlington, VA, June 2001, pp. 50–55.
- ²Ryan, A., Zennaro, M., Howell, A., Sengupta, R., and Hedrick, J. K., "An overview of emerging results in cooperative UAV control," *IEEE Conf. on Decision and Control*, 2004.
- ³Dubins, L. E., "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents," *American Journal of Mathematics*, Vol. 79, 1957, pp. 497–516.
- ⁴McGee, T. G., Spry, S., and Hedrick, J. K., "Optimal path planning in a constant wind with a bounded turning rate," *AIAA Conf. on Guidance, Navigation and Control*, San Francisco, CA, Aug. 2005, Electronic Proceedings.
- ⁵Gutin, G. and Punnen, A. P., *The Traveling Salesman Problem and Its Variations*, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1st ed., 2002.
- ⁶Obermeyer, K. J., "Path Planning for a UAV Performing Reconnaissance of Static Ground Targets in Terrain," *AIAA Conf. on Guidance, Navigation and Control*, Chicago, IL, Aug. 2009, To appear.
- ⁷Betts, J. T., "Survey of Numerical Methods for Trajectory Optimization," *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 21, No. 2, 1998, pp. 193–207.
- ⁸Papadimitriou, C. H., "The Euclidean Traveling Salesman Problem is NP-Complete," *Theoretical Computer Science*, Vol. 4, 1977, pp. 237–244.
- ⁹de Berg, M., Gudmundsson, J., Katz, M. J., Levkopoulos, C., Overmars, M. H., and van der Stappen, A. F., "TSP with Neighborhoods of Varying Size," 2002, pp. 21–35.
- ¹⁰Dumitresco, A. and Mitchell, J. S. B., "Approximation Algorithms for TSP with Neighborhoods in the Plane," *Journal of Algorithms*, Vol. 48, No. 1, 2003, pp. 135–159.
- ¹¹Mata, C. S. and Mitchell, J. S. B., "Approximation Algorithms for Geometric Tour and Network Design Problems," *Symposium on Computational Geometry*, 1995, pp. 360–369.
- ¹²Le Ny, J., Frazzoli, E., and Feron, E., "The curvature-constrained traveling salesman problem for high point densities," *IEEE Conf. on Decision and Control*, 2007, pp. 5985–5990.
- ¹³Nygaard, K. E., Chandler, P. R., and Pachter, M., "Dynamic network flow optimization models for air vehicle resource allocation," 2001.
- ¹⁴Schumacher, C., Chandler, P. R., and Rasmussen, S. R., "Task allocation for wide area search munitions," *American Control Conference*, 2002.
- ¹⁵Tang, Z. and Özgüner, Ü., "Motion Planning for Multi-Target Surveillance with Mobile Sensor Agents," *IEEE Transactions on Robotics*, Vol. 21, No. 5, 2005, pp. 898–908.
- ¹⁶Rathinam, S., Sengupta, R., and Darbha, S., "A Resource Allocation Algorithm for Multi-Vehicle Systems with Non holonomic Constraints," *IEEE Transactions on Automation Sciences and Engineering*, Vol. 4, No. 1, 2007, pp. 98–104.
- ¹⁷Savla, K., Frazzoli, E., and Bullo, F., "Traveling Salesperson Problems for the Dubins vehicle," *IEEE Transactions on Automatic Control*, Vol. 53, No. 6, 2008, pp. 1378–1391.
- ¹⁸Le Ny, J. and Feron, E., "An Approximation Algorithm for the Curvature-Constrained Traveling Salesman Problem," *Allerton Conf. on Communications, Control and Computing*, Monticello, IL, USA, Sept. 2005.
- ¹⁹Oberlin, P., Rathinam, S., and Darbha, S., "A Transformation for a Heterogeneous, Multiple Depot, Multiple Traveling Salesmen Problem," *American Control Conference*, 2009, pp. 1292–1297.
- ²⁰Yadlapalli, S., Malik, W. A., Darbha, S., and Pachter, M., "A Lagrangian-based algorithm for a Multiple Depot, Multiple Traveling Salesmen Problem," *Nonlinear Analysis: Real World Applications*, Vol. 10, No. 4, August 2009, pp. 1990–1999.
- ²¹Noon, C. E. and Bean, J. C., "An Efficient Transformation of the Generalized Traveling Salesman Problem," Tech. Rep. 91-26, Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, 1991.
- ²²Fischetti, M., Salazar-González, J. J., and Toth, P., *The Traveling Salesman Problem and its Variations*, chap. 13, Kluwer, 2002.
- ²³Choset, H., Lynch, K. M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L. E., and Thrun, S., *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, 2005.
- ²⁴LaValle, S. M., *Planning Algorithms*, Cambridge University Press, 2006, Available at <http://planning.cs.uiuc.edu>.
- ²⁵Ceccarelli, N., Enright, J. J., Frazzoli, E., Rasmussen, S. J., and Schumacher, C. J., "Micro UAV Path Planning for Reconnaissance in Wind," *American Control Conference*, New York, NY, USA, July 2007, pp. 5310–5315.
- ²⁶Latombe, J.-C., *Robot Motion Planning*, Kluwer Academic Publishers, 1991.
- ²⁷Boissonnat, J.-D., Cérézo, A., and Leblond, J., "Shortest paths of bounded curvature in the plane," *Journal of Intelligent and Robotic Systems*, Vol. 11, 1994, pp. 5–20.
- ²⁸O'Rourke, J., *Computational Geometry in C*, Cambridge University Press, 2000.
- ²⁹Helsgaun, K., "An effective implementation of the LinKernighan traveling salesman heuristic," *European Journal of Operational Research*, Vol. 126, No. 1, October 2000, pp. 106–130.
- ³⁰Applegate, D. L., Bixby, R. E., and Chvátal, V., *The Traveling Salesman Problem: A Computational Study*, Applied Mathematics Series, Princeton University Press, 2006.
- ³¹Lin, S. and Kernighan, B. W., "An effective heuristic algorithm for the traveling-salesman problem," *Operations Research*, Vol. 21, 1973, pp. 498–516.
- ³²Bui, X. N., Boissonnat, J. D., Soueres, P., and Laumond, J. P., "Shortest Path Synthesis for Dubins Non-Holonomic Robot," *IEEE Transactions on Robotics and Automation*, 1994.
- ³³Goodman, J. E. and O'Rourke, J., editors, *Handbook of Discrete and Computational Geometry*, CRC Press, 2nd ed., 2004.

- ³⁴Fulton, W., *Intersection Theory*, Ergebnisse der Mathematik und ihrer Grenzgebiete, Springer Verlag, Berlin, Heidelberg, New York, 1984.
- ³⁵Niederreiter, H., *Random Number Generation and Quasi-Monte Carlo Methods*, No. 63 in CBMS-NSF Regional Conference Series in Applied Mathematics, Society for Industrial & Applied Mathematics, 1992.
- ³⁶Karaman, S. and Frazzoli, E., “Incremental Sampling-based Algorithms for Optimal Motion Planning,” *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, jun 2010.
- ³⁷Applegate, D., Bixby, R., Chvátal, V., and Cook, W., “On the solution of traveling salesman problems,” *Documenta Mathematica, Journal der Deutschen Mathematiker-Vereinigung*, Berlin, Germany, Aug. 1998, pp. 645–656, Proceedings of the International Congress of Mathematicians, Extra Volume ICM III.
- ³⁸Kreĭn, M. G. and Nudelman, A. A., *The Markov moment problem and extremal problems: ideas and problems of P. L. Čebyšev and A. A. Markov and their further development*, American Mathematical Society, 1977.
- ³⁹McNeely, R., Iyer, R. V., and Chandler, P., “Tour Planning for an Unmanned Air Vehicle under Wind Conditions,” *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 30, No. 5, 2007, pp. 1299–1306.
- ⁴⁰Techy, L. and Woolsey, C. A., “Minimum-Time Path Planning for Unmanned Aerial Vehicles in Steady Uniform Winds,” *AIAA Journal of Guidance, Control, and Dynamics*, Vol. 32, No. 6, 2009, pp. 1736–1746.
- ⁴¹Hershberger, J. and Suri, S., “An Optimal Algorithm for Euclidean Shortest Paths in the Plane,” *SIAM Journal on Computing*, Vol. 28, 1999, pp. 2215–2256.
- ⁴²Reif, J. and Wang, H., “The Complexity of the Two Dimensional Curvature-Constrained Shortest-Path Problem,” *In Proc. Third International Workshop on the Algorithmic Foundations of Robotics*, 1998, pp. 49–57.
- ⁴³Eele, A. and Richards, A., “Path-Planning with Avoidance using Nonlinear Branch-and-Bound Optimisation,” *AIAA Conf. on Guidance, Navigation and Control*, Hilton Head, South Carolina, 2007, Electronic Proceedings.
- ⁴⁴Eele, A. and Richards, A., “Comparison of Branching Strategies for Path-Planning with Avoidance using Nonlinear Branch-and-Bound,” *AIAA Conf. on Guidance, Navigation and Control*, Honolulu, Hawaii, 2008, Electronic Proceedings.
- ⁴⁵Borrelli, F., Subramanian, D., Raghunathan, A. U., and Biegler, L. T., “MILP and NLP Techniques for centralized trajectory planning of multiple unmanned air vehicles,” *American Control Conference*, 2006.
- ⁴⁶Richards, A. and How, J. P., “Aircraft trajectory planning with collision avoidance using mixed integer linear programming,” *American Control Conference*, 2002, pp. 1936–1941.

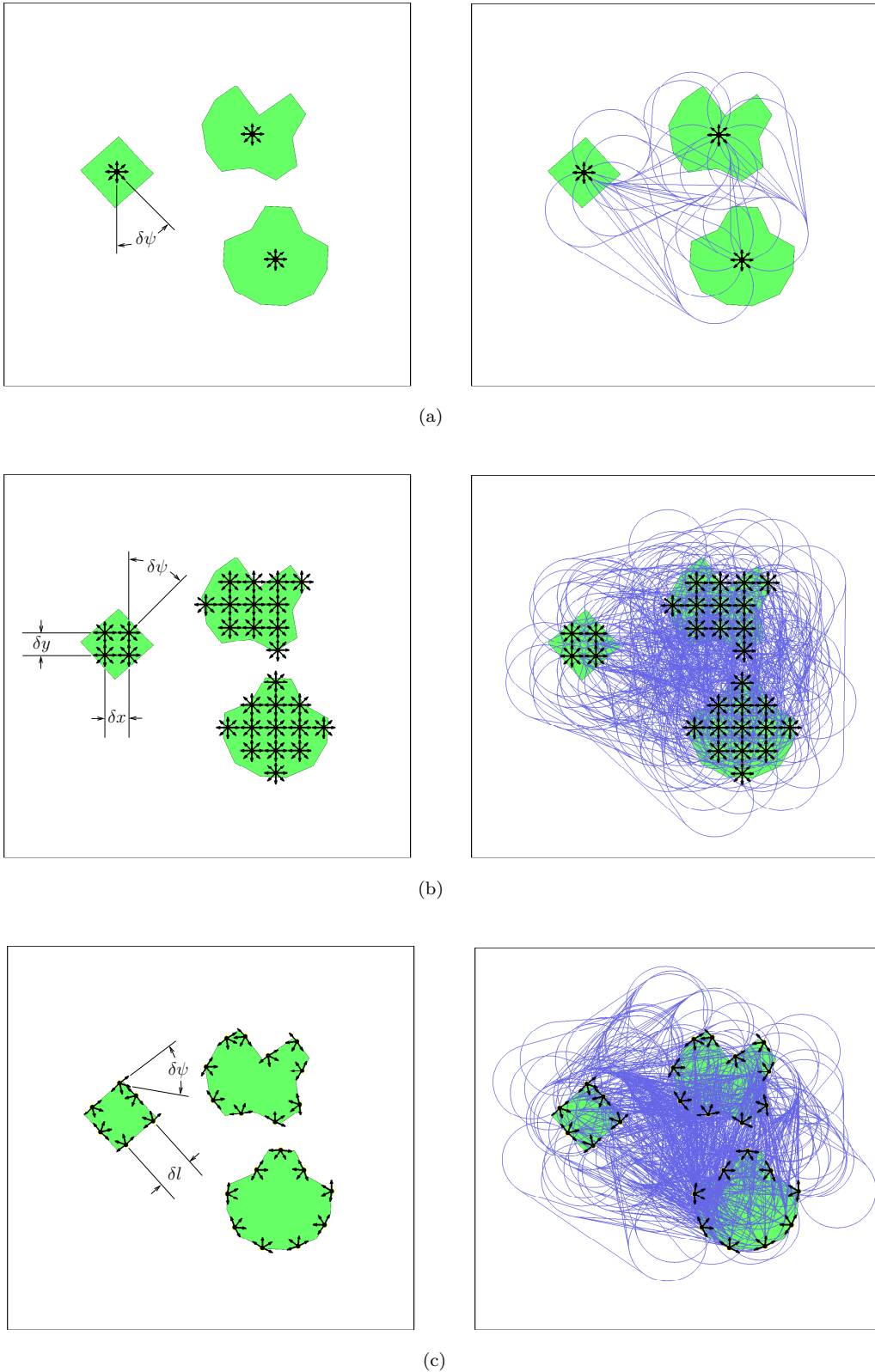


Figure 4. Roadmap constructions for (a) the DTSP through target positions, (b) the PVDTSP with interior pose sampling, and (c) the PVDTSP with entry pose sampling. Sampled poses are represented by black arrows. In each case, a uniform grid is first constructed according to specified spacings (left), then the grid samples are connected by minimum-time Dubins paths (blue curves, right). To reduce clutter, only a representative subset of all connections are shown in (b) and (c). The best tours extracted from these roadmaps are shown in Fig. 5.

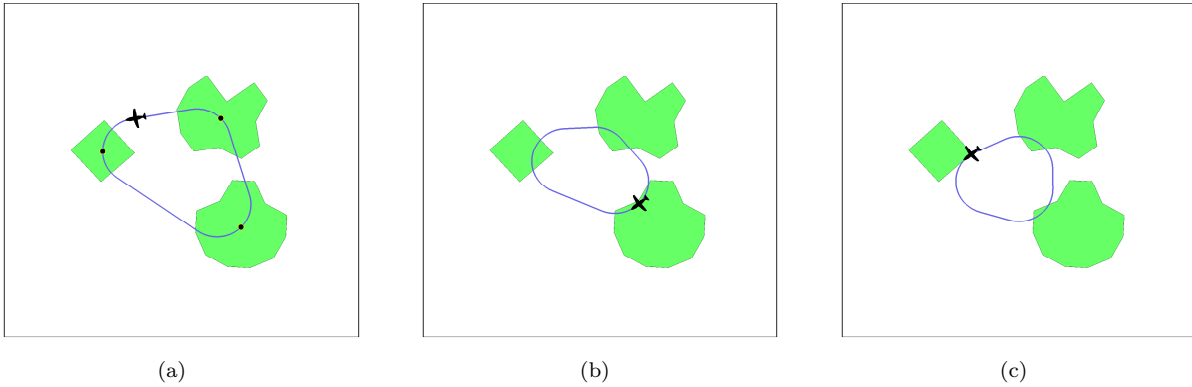


Figure 5. The best tours extracted from the roadmaps shown in Fig. 4a, b, and c, respectively. In (a) the DTSP tour is restricted to pass directly over the target locations (black disks), but the PVDTS tours in (b) and (c) exploit the freedom of the visibility sets.

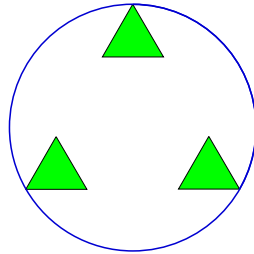


Figure 6. An example of a PVDTS instance which is both interior pose and entry pose degenerate. The globally optimal solution τ^* is a minimum turn radius circle (blue) which just grazes the outside corners of the triangular visibility regions. Any tour $\tau \neq \tau^*$ would have to veer away from the visibility regions and then come back in order to visit them all. The cost $C(\tau)$ is thus bounded away from $C(\tau^*)$.

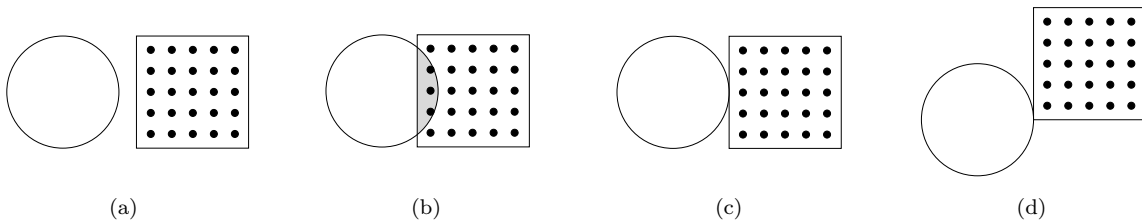
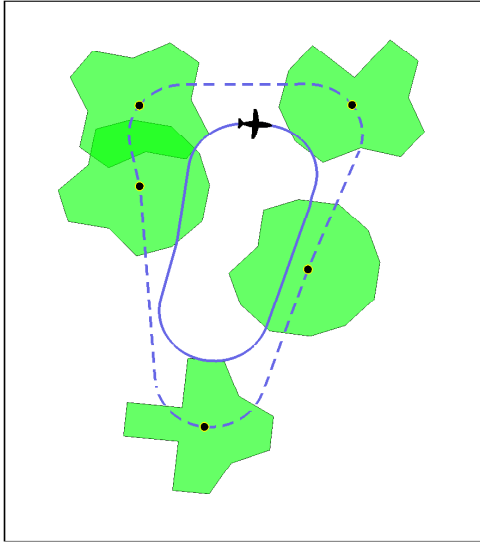


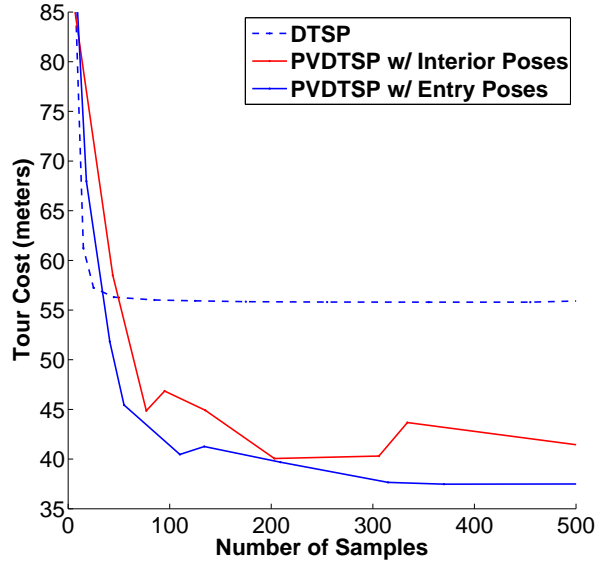
Figure 7. A grid sampling strategy is used to minimize the function f in Eq. 7 over a square. The performance of the strategy depends on the relative configurations of the unit disk and square. If (a) the disk and square don't intersect or (b) their intersection contains an open set, then the value returned by the sampling strategy is guaranteed to approach the optimum as the grid becomes finer. If, however, the problem is degenerate due to (c) tangency or (d) corner point intersection, then the sampling strategy will not find the optimum no matter how fine the grid is.

Tours from DTSP (--) and PVDTSP (-) Algorithms



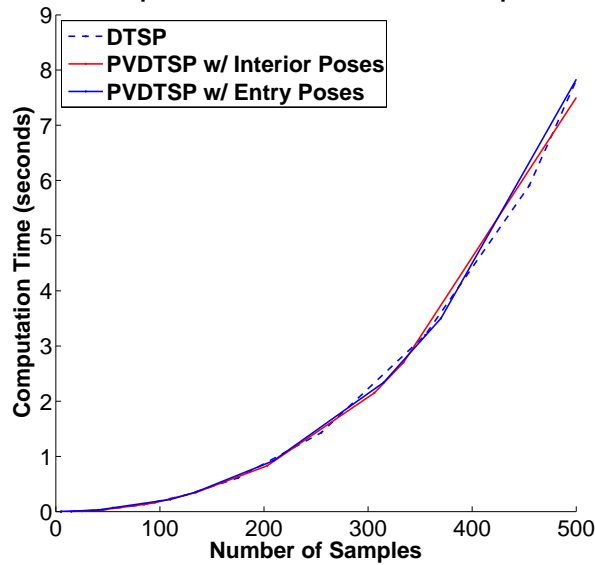
(a)

Tour Cost vs. Number of Samples



(b)

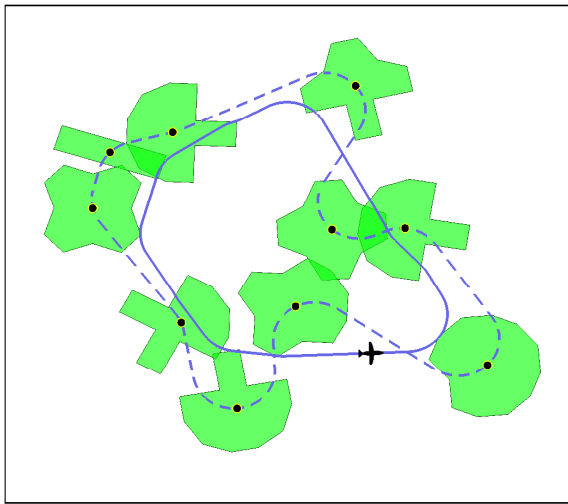
Computation Time vs. Number of Samples



(c)

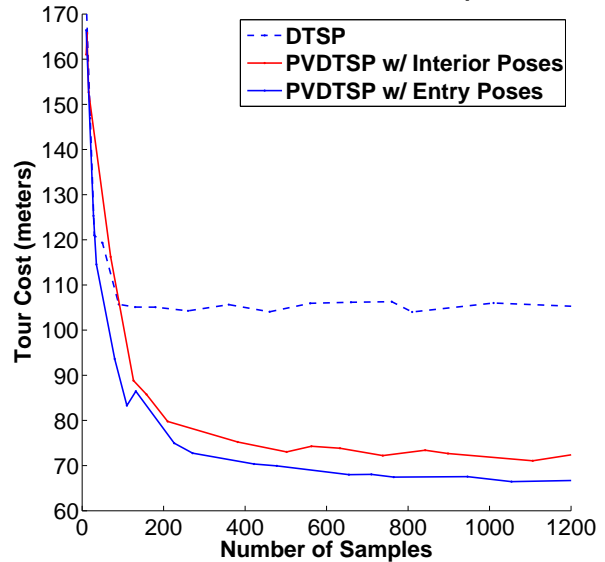
Figure 8. Example PVDTSP instance with $n = 5$ targets and aircraft minimum turn radius $r_{\min} = 3$ m. Solutions were computed using the sampling-based DTSP solver from Ref. 19, PVDTSP_SOLVE with INTERIOR_POSE_ROADMAP ($\alpha = 2.2$), and PVDTSP_SOLVE with ENTRY_POSE_ROADMAP ($\alpha = 2.85$). (a) Black dots represent target locations. Green polygons represent the target visibility regions. The dashed blue curve shows the best tour found by the DTSP solver. The solid blue curve shows the best tour found by PVDTSP_SOLVE using ENTRY_POSE_ROADMAP. The best solution found using INTERIOR_POSE_ROADMAP is nearly identical to that found using ENTRY_POSE_ROADMAP and hence is not shown. (b,c) Tour costs and computation times for each of the algorithms over a range of sample counts. Cf. Table 5.

Tours from DTSP (---) and PVDTSP (-) Algorithms



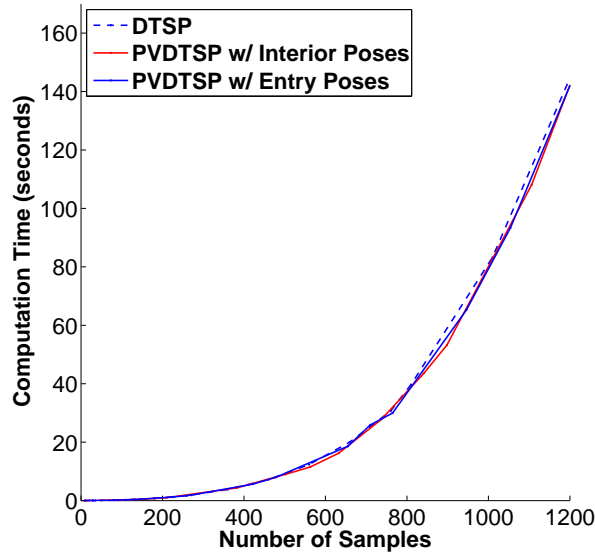
(a)

Tour Cost vs. Number of Samples



(b)

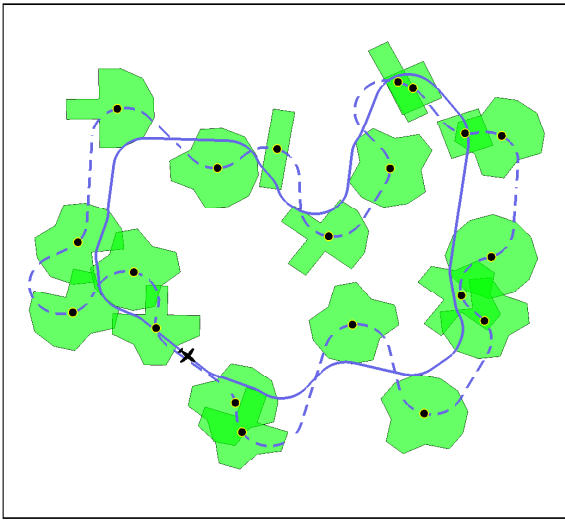
Computation Time vs. Number of Samples



(c)

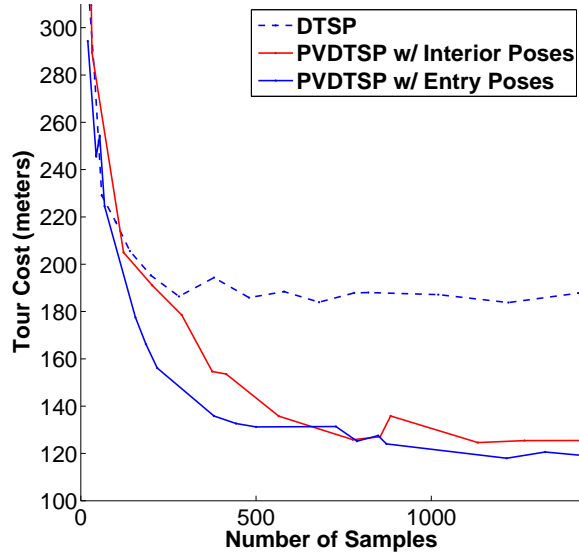
Figure 9. Example PVDTSP instance with $n = 10$ targets and aircraft minimum turn radius $r_{\min} = 3$ m. Solutions were computed using the DTSP solver from Ref. 19, PVDTSP_SOLVE with INTERIOR_POSE_ROADMAP ($\alpha = 2.2$), and PVDTSP_SOLVE with ENTRY_POSE_ROADMAP ($\alpha = 2.85$). (a) Black dots represent target locations. Green polygons represent the target visibility regions. The dashed blue curve shows the best tour found by the DTSP solver. The solid blue curve shows the best tour found by PVDTSP_SOLVE using ENTRY_POSE_ROADMAP. The best solution found using INTERIOR_POSE_ROADMAP is nearly identical to that found using ENTRY_POSE_ROADMAP and hence is not shown. (b,c) Tour costs and computation times for each of the algorithms over a range of sample counts. Cf. Table 5.

Tours from DTSP (--) and PVDTSP (-) Algorithms



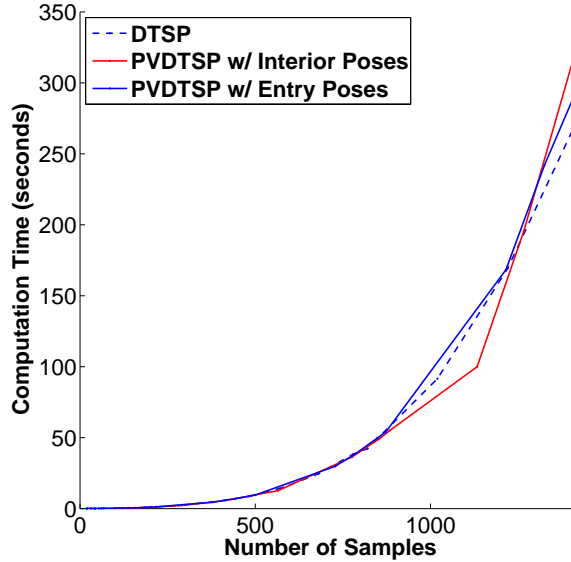
(a)

Tour Cost vs. Number of Samples



(b)

Computation Time vs. Number of Samples



(c)

Figure 10. Example PVDTSP instance with $n = 20$ targets and aircraft minimum turn radius $r_{\min} = 3$ m. Solutions were computed using the DTSP solver from Ref. 19, PVDTSP_SOLVE with INTERIOR_POSE_ROADMAP ($\alpha = 2.2$), and PVDTSP_SOLVE with ENTRY_POSE_ROADMAP ($\alpha = 2.85$). (a) Black dots represent target locations. Green polygons represent the target visibility regions. The dashed blue curve shows the best tour found by the DTSP solver. The solid blue curve shows the best tour found by PVDTSP_SOLVE using ENTRY_POSE_ROADMAP. The best solution found using INTERIOR_POSE_ROADMAP is nearly identical to that found using ENTRY_POSE_ROADMAP and hence is not shown. (b,c) Tour costs and computation times for each of the algorithms over a range of sample counts. Cf. Table 5.