Path Planning for a UAV Performing Reconnaissance of Static Ground Targets in Terrain *

Karl J. Obermeyer†

In this article we consider a path planning problem for a single fixed-wing aircraft performing an ISR (Intelligence Surveillance Reconnaissance) mission using EO (Electro-Optical) camera(s). We give a mathematical formulation of the general aircraft visual reconnaissance problem for static ground targets in terrain and show that, under simplifying assumptions, it can be reduced to a variant of the Traveling Salesman Probem which we call the PVDTSP (Polygon-Visiting Dubins Traveling Salesman Problem). We design a genetic algorithm to solve the PVDTSP and validate it in a Monte Carlo numerical study. For fixed computation time, the genetic algorithm produces reconnaissance tours on average nearly half the length (and thus can be flown in half the time) of those delivered by a random search. The modular design of the genetic algorithm allows it to easily be extended to handle realistic assumptions such as wind and airspace constraints.

I. Introduction

In this article we consider a path planning problem for a single fixed-wing aircraft performing an ISR (Intelligence Surveillance Reconnaissance) mission using EO (Electro-Optical) camera(s). Such missions are relevant to both civil and military UAV systems, e.g., the US Air Force's COUNTER (Coopertive Operations in UrbaN TERain) program.[1,2] Given a set of stationary ground targets in a terrain (natural, urban, or mixed), our 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 (i) the target is in close enough range to satisfy the photograph's resolution requirements, and (ii) 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, aircraft path planning can be complicated by wind, airspace constraints (e.g. due to enemy threats and 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] and approximate the targets' visibility regions by polygons, the reconnaissance path planning problem can be reduced to the following.

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

We refer to this henceforth as the *PVDTSP* (for *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

*This version: March 19, 2009

This work was supported by a DoD SMART fellowship (`http://www.asee.org/fellowships/smart/`). Thanks to the following people for helpful comments: F. Bullo (UCSB), S. Darbha (Tex. A&M), R. Holsapple (WPAFB), D. Kingston (WPAFB), M. Mears (WPAFB), P. Oberlin (Tex. A&M), S. Rasmussen (Miami Valley Aerospace LLC), C. Schumacher (WPAFB), V. Shaferman (Technion)

†K. Obermeyer is a PhD student at the Center for Control, Dynamical Systems, and Computation, University of California at Santa Barbara, Santa Barbara, CA 93106, USA; `karl@engr.ucsb.edu`

[a] A Dubins vehicle is one which moves only forward and has a minimum turning radius.[3,4]

[b] The Traveling Salesman Problem, one of the most famous NP-hard problems of combinatorial optimization, is to find the shortest Hamiltonian cyle (visits every node exactly once) in a weighted graph. See, e.g.,[5] for a brief introduction, or[6] for an extensive treatment

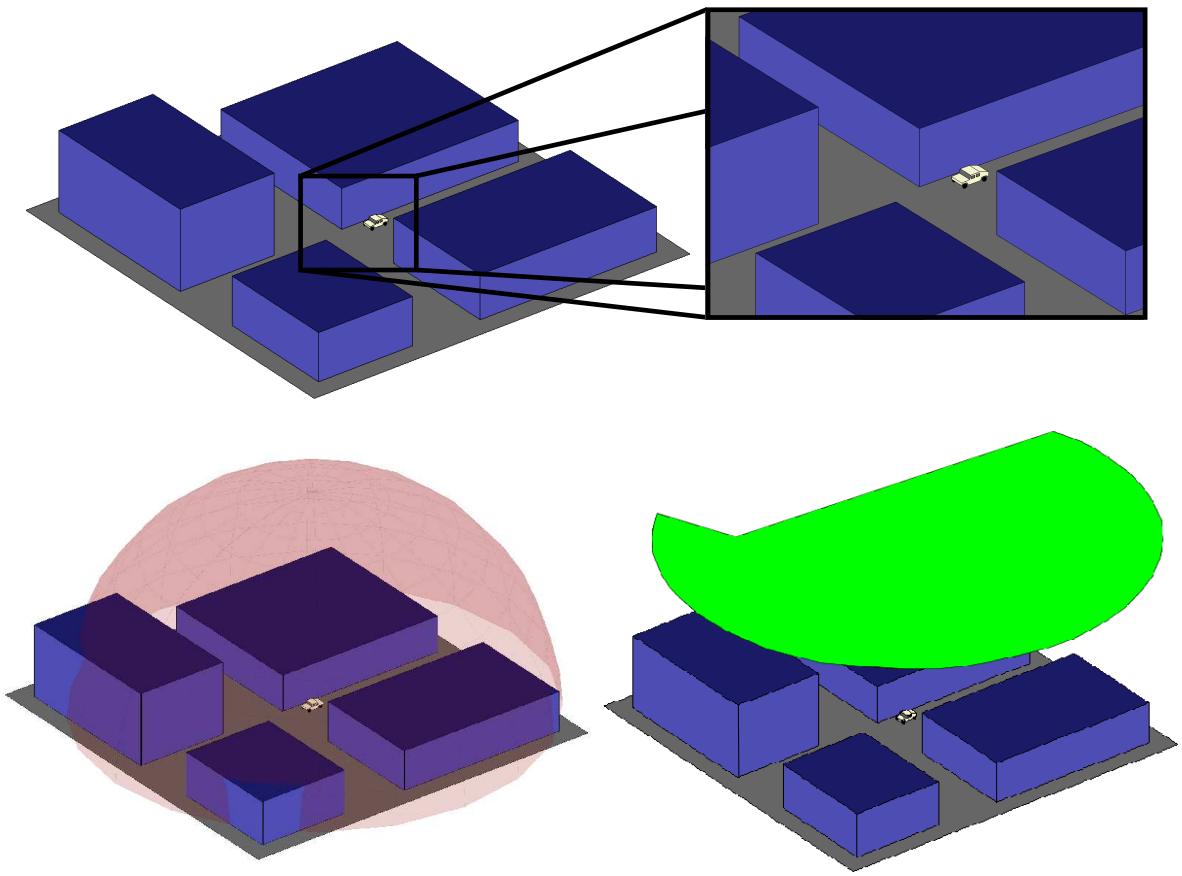


Figure 1. Top is shown an example target, a ground vehicle parked next to a building in urban terrain. The set of all points which are close enough to the target to satisfy photograph resolution requirements is a solid sphere (bottom left). The green two-dimensional region in the sky (bottom right) shows 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.

is shown in Fig. 2.

The DTSP (Dubins TSP for a vehicle visiting single point targets) is known to be NP-hard in the number n of targets.[7] Constant factor approximation algorithms exist for the DTSP, e.g., the "Alternating Algorithm" of Ref. 8 computes a DTSP approximate solution using the optimal solution to the underlying Euclidean TSP, or Ref. 9 produces a DTSP approximate solution in time $\mathcal{O}(n^3)$. To our knowledge the PVDTSP has not previously been studied and it is not clear how to extend existing DTSP algorithms to exploit the extra degrees of freedom in being able to choose which points in the target polygons a PVDTSP tour visits. Since the TSP and most of its variations are NP-Hard, one possible approach to obtaining good solutions in reasonable computation time is to use metaheuristic algorithms such as tabu search, simulated annealing, or genetic algorithms.[10] A genetic algorithm, which we apply to the PVDTSP in the present work, is an umbrella term referring to any iterative procedure which mimics biological evolution by operating on a population of candidate solutions encoded as so-called chromosomes. The genetic operators of crossover and mutation are successively applied, generation after generation, until a sufficiently fit solution appears in the population. Genetic algorithms have recently been applied to variations of the TSP as well as UAV motion planning problems, however it is not obvious how to adapt these formulations to the PVDTSP, nor is it clear whether any such adaptations would be effective.[11–17]

There are three 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 show this general form reduces to the PVDTSP. As a second contribution, we design a genetic algorithm for the PVDTSP and validate it with a Monte Carlo numerical study. Third, we describe how the modular nature of our genetic algorithm allows it to easily be extended to handle more realistic assumptions such as wind

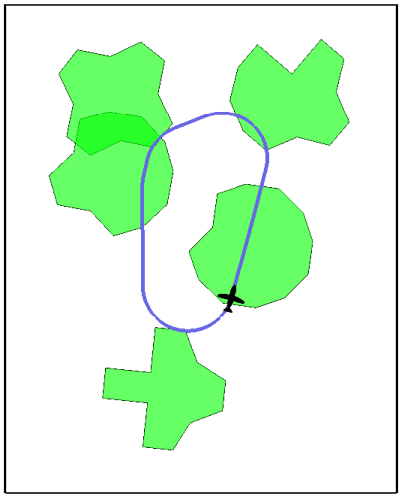


Figure 2. Example problem instance and candidate solution path for the *PVDTSP* **(***Polygon-Visiting Dubins Traveling Salesman Problem***). In order to photograph all targets, the aircraft must fly through at least one point in each target's visibility region (green), cf. Fig. 1.**

and airspace constraints.

This article is organized as follows. In Sec. II we introduce some notation and a precise mathematical formulation of the minimum time aircraft path planning problem. In Sec. III we design the genetic algorithm which is validated by a numerical study in Sec. IV. Finally, we describe how to extend the genetic algorithm in Sec. V, then conclude in Section VI.

II. Mathematical Formulation

We begin with some preliminary notation. Let $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n\}$ be the set of n targets which must be photographed by our aircraft. Given a set A, we denote the power set of A, 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 complete state of our reconnaissance aircraft is encoded in a vector $\mathbf{x}$, which takes a value in the aircraft's state space X. We can segregate $\mathbf{x}$ into internal and external states so that

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{\text{internal}} \\ \mathbf{x}_{\text{external}} \end{bmatrix} \in X = X_{\text{internal}} \times X_{\text{external}}. \tag{1}$$

The internal state $\mathbf{x}_{\text{internal}}$ accounts for control surface configurations, and more importantly, if the aircraft has gimbaled camera(s), then also for the camera configuration(s). The external state $\mathbf{x}_{\text{external}}$ accounts for the aircraft body position and velocity in the full six degrees of freedom.

We now define a map $\mathcal{V} : \mathcal{T} \to 2^X$ from the set of targets to subsets of the aircraft state space. Under this map, $\mathcal{V}(\mathcal{T}_i) \subset X$, called the ith 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. Later, in Sec. II.A, we discuss how to calculate such regions from a terrain model, but let us assume for now we can make this calculation. We also assume the ability to calculate 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 **minimimum time reconnaissance path planning problem** can be stated as follows

$$\begin{array}{rl} \text{Minimize :} & C \;=\; \sum_{i=1}^{n-1} d(\mathbf{x}_i, \mathbf{x}_{i+1}) + d(\mathbf{x}_n, \mathbf{x}_1) \\ \text{Subject To :} & \forall i \in \{1, \ldots, n\}\ \exists j \in \{1, \ldots, n\} \text{ s.t. } \mathbf{x}_j \in \mathcal{V}(\mathcal{T}_i), \end{array} \tag{2}$$

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

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

(i) The aircraft is modeled as a Dubins vehicle with minimum turning radius $\rho_{\min}$, fixed altitude h, and constant airspeed V.
Comments: Common for small low-power UAVs.

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

(iii) There are no airspace constraints nor wind.
Comments: As to be discussed in Sec. V, our results can easily be extended to handle wind and no-fly zones.

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

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \end{bmatrix} = \begin{bmatrix} V\sin(\psi) \\ V\cos(\psi) \\ u \end{bmatrix}, \tag{3}$$

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 $\mathbf{x}_{\text{internal}}$ so that the aircraft state space is reduced to

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

and the Visibility sets $\mathcal{V}(\mathcal{T}_1), \ldots, \mathcal{V}(\mathcal{T}_n)$ are reduced to 2-dimensional regions in $\mathbb{R}^2$ as shown in Fig. 1 and 2 (as opposed to subsets of $X = \mathbb{R}^2 \times \mathbb{S}$). The minimum time path between two Dubins states $\mathbf{x}$ and $\mathbf{x}'$ can be computed very quickly and in constant time.[3,4] This provides us with our "black box" distance function $d(\mathbf{x}, \mathbf{x}')$ as it appears in the optimization problem Eq. 2. Although visibility regions may contain circular arcs due to the camera range constraint, they can be well approximated by polygons. We have now reduced our minimum time reconnaissance path planning problem to a PVDTSP.

II.A. Calculating Visibility Regions

In order to calculate the visibility region $\mathcal{V}(\mathcal{T}_i)$ of a target, it is necessary to know the target location and to have a computer model/representation of the terrain. This representation may be either a vector format, e.g., a TIN (Triangulated Irregular Network), or a raster format, e.g., regular-grid DEM (Digital Elevation Model) or the military's DTED (Digital Terrain Elevation Data). The necessary data to build a terrain model could be gathered, e.g., by LIDAR (LIght Detection And Ranging), SAR (Synthetic Aperture Radar), or photogrammetry. Once the terrain model has been built, the visibility region of a target may be calculated using Bresenham's line algorithm[18] in the raster case, or a "sweeping algorithm"[16,17,19] in the vector case.

III. Solution by Genetic Algorithm

In this section we present a genetic algorithm which will deliver a quick feasible solution to the PVDTSP with monotonic improvement over runtime. Later, in Sec. IV, we validate this algorithm with a numerical study, then in Sec. V show that it can be extended to handle wind and airspace constraints. For details on genetic algorithms in general, we suggest Refs. 20 and 10.

The first step in designing a genetic algorithm is to decide on an encoding to represent candidate solutions. Each encoded solution will be a *chromosome* in the *population* which evolves over the course of the genetic algorithm. For our encoding we use a sequence of doubles

$$((\mathcal{T}_{k_1}, \mathbf{x}_1), (\mathcal{T}_{k_2}, \mathbf{x}_2), (\mathcal{T}_{k_3}, \mathbf{x}_3), \ldots, (\mathcal{T}_{k_n}, \mathbf{x}_n)), \tag{4}$$

where we refer to a double $(\mathcal{T}_{k_i}, \mathbf{x}_i)$ as a *node* of the tour. The sequence $k_1, \ldots, k_n$ represents a permutation of the target identifiers $1, \ldots, n$. This is the order in which the aircraft will visit the targets. Each aircraft

state $\mathbf{x}_i$ is in the k_ith target's visibility set $\mathcal{V}(\mathcal{T}_{k_i})$. If a solution is accepted, the aircraft flies to each state $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in the order they appear in the chromosome, photographing $\mathcal{T}_{k_i}$ when it reaches $\mathbf{x}_i$. Shortest Dubins paths are flown between successive states in the sequence. We define the *fitness* of a chromosome c to be $f(c) = 1/C(c)$, where C is the cost shown in Eq. 2. We say a chromosome c is more *fit* than another c' if $f(c) > f(c')$.

Referring to the pseudocode in Tab. 1, we now describe how our genetic algorithm operates. The algorithm has five fixed parameters: *populations size* N, *crossover probability* p_c, *mutation probability* p_m, *elite group size* N_e, and *number of generations* N_g. The population P is initialized to a set of N random chromosomes. A random chromosome is produced by randomly shuffling the integers $1, \ldots, n$, uniform randomly sampling points in the the targets' visibility sets (for the state positions), and uniform randomly sampling angles on the interval $[0, 2\pi)$ (for state orientations). Now we enter the main loop (line 2). At the beginning of the main loop, the N_e fittest chromosomes are copied into the next generation population P'. This ensures the fitness of the fittest chomosome in the P can never decrease during the execution of the algorithm. Next, in order to make P' have size N, we need to produce $N - N_e - 1$ offspring. Offspring are produced two at a time as follows. Two chromosomes, c_{mom} and c_{dad} are selected randomly from P with probability proportional to their respective fitnesses. This is known as *roulette wheel* parent selection. With probability p_c, children c_i and c_{i+1} are generated by crossing over the parents, otherwise they are just copies of the parents (see Sec. III.A below for a detailed description of our crossover operation). Once the children are in place, we randomly perform mutation operators on them with probabilities p_m, p_m, $p_m/7$, and $p_m/7$, respectively[c] (see Sec. III.B below for a detailed description of our mutation operators). Once all the children have been constructed and P' has size n, P and P' are swaped, i.e. the new generation replaces the old. The process continues for N_g generations. After termination, the algorithm returns the fittest chromosome in the population, which represents the shortest aircraft tour found.

Table 1. Genetic Algorithm for Minimizing Aircraft Roconnaissance Tour Length

```
1: construct random initial population P = {c_1, c_2, ..., c_N} of N chromosomes;
2: for all generations 1, 2, 3, ..., N_g do
3:    copy N_e best chromosomes from P directly into new population P';
4:    for all i = 1, 3, 5, 7, ..., N − N_e − 1 do
5:       roulette wheel select parents c_mom and c_dad from P;
6:       generate children c'_i and c'_{i+1} by crossover with probability p_c,
         otherwise c'_i ← c_mom and c'_{i+1} ← c_dad;
7:       for all j = i, i + 1 do
8:          orientation shift mutate c'_j with probability p_m;
9:          position shift mutate c'_j with probability p_m;
10:         swap mutate c'_j with probability p_m/7.0;
11:         partial reverse mutate c'_j with probability p_m/7.0;
12:      P' = P' ∪ {c'_i, c'_{i+1}};
13:   swap(P, P');
14: return best chromosome in P;
```

III.A. Crossover

Our crossover operator, which appears in line 6 of the pseudocode Tab. 1, produces offspring by preserving partial tours from the parents. We have adapted for our problem the so-called Order Crossover (OX)[10] used for the TSP. An offspring is constructed by choosing a node subsequence from one parent, then filling in the remaining nodes in the order they appear in the other parent. We illustrate by and example having $n = 9$ targets. Let the parents be

$$c_{\text{mom}} = ((\mathcal{T}_2, \mathbf{x}_1), (\mathcal{T}_3, \mathbf{x}_2), (\mathcal{T}_5, \mathbf{x}_3), (\mathcal{T}_7, \mathbf{x}_4), (\mathcal{T}_4, \mathbf{x}_5), (\mathcal{T}_6, \mathbf{x}_6), (\mathcal{T}_9, \mathbf{x}_7), (\mathcal{T}_8, \mathbf{x}_8), (\mathcal{T}_1, \mathbf{x}_9))$$

$$c_{\text{dad}} = ((\mathcal{T}_6, \tilde{\mathbf{x}}_1), (\mathcal{T}_1, \tilde{\mathbf{x}}_2), (\mathcal{T}_7, \tilde{\mathbf{x}}_3), (\mathcal{T}_9, \tilde{\mathbf{x}}_4), (\mathcal{T}_3, \tilde{\mathbf{x}}_5), (\mathcal{T}_5, \tilde{\mathbf{x}}_6), (\mathcal{T}_8, \tilde{\mathbf{x}}_7), (\mathcal{T}_4, \tilde{\mathbf{x}}_8), (\mathcal{T}_2, \tilde{\mathbf{x}}_9)).$$

[c]The last two mutation probabilities were scaled by 1/7 because it seems to give better performance.

First, two cut points are uniform randomly chosen, reperesented by the bars |,

$$c_{\text{mom}} = ((\mathcal{T}_2, \mathbf{x}_1), (\mathcal{T}_3, \mathbf{x}_2) \mid (\mathcal{T}_5, \mathbf{x}_3), (\mathcal{T}_7, \mathbf{x}_4), (\mathcal{T}_4, \mathbf{x}_5), (\mathcal{T}_6, \mathbf{x}_6) \mid (\mathcal{T}_9, \mathbf{x}_7), (\mathcal{T}_8, \mathbf{x}_8), (\mathcal{T}_1, \mathbf{x}_9))$$

$$c_{\text{dad}} = ((\mathcal{T}_6, \tilde{\mathbf{x}}_1), (\mathcal{T}_1, \tilde{\mathbf{x}}_2) \mid (\mathcal{T}_7, \tilde{\mathbf{x}}_3), (\mathcal{T}_9, \tilde{\mathbf{x}}_4), (\mathcal{T}_3, \tilde{\mathbf{x}}_5), (\mathcal{T}_5, \tilde{\mathbf{x}}_6) \mid (\mathcal{T}_8, \tilde{\mathbf{x}}_7), (\mathcal{T}_4, \tilde{\mathbf{x}}_8), (\mathcal{T}_2, \tilde{\mathbf{x}}_9)).$$

Next, the sections between the cut points are placed directly into the offspring,

$$c_i = (\ -\ ,\ -\ \mid (\mathcal{T}_5, \mathbf{x}_3), (\mathcal{T}_7, \mathbf{x}_4), (\mathcal{T}_4, \mathbf{x}_5), (\mathcal{T}_6, \mathbf{x}_6) \mid -\ ,\ -\ ,\ -\ )$$

$$c_{i+1} = (\ -\ ,\ -\ \mid (\mathcal{T}_7, \tilde{\mathbf{x}}_3), (\mathcal{T}_9, \tilde{\mathbf{x}}_4), (\mathcal{T}_3, \tilde{\mathbf{x}}_5), (\mathcal{T}_5, \tilde{\mathbf{x}}_6) \mid -\ ,\ -\ ,\ -\ ).$$

It now remains to fill in the blank spaces $(-)$ in the offspring. The rest of c_i is completed as follows. The order of nodes in c_{dad}, starting from the second cut point, is

$$((\mathcal{T}_8, \tilde{\mathbf{x}}_7), (\mathcal{T}_4, \tilde{\mathbf{x}}_8), (\mathcal{T}_2, \tilde{\mathbf{x}}_9), (\mathcal{T}_6, \tilde{\mathbf{x}}_1), (\mathcal{T}_1, \tilde{\mathbf{x}}_2), (\mathcal{T}_7, \tilde{\mathbf{x}}_3), (\mathcal{T}_9, \tilde{\mathbf{x}}_4), (\mathcal{T}_3, \tilde{\mathbf{x}}_5), (\mathcal{T}_5, \tilde{\mathbf{x}}_6)).$$

Deleting from this ordering the nodes with targets $\{\mathcal{T}_5, \mathcal{T}_7, \mathcal{T}_4, \mathcal{T}_6\}$ already present in the first offspring, we obtain

$$((\mathcal{T}_8, \tilde{\mathbf{x}}_7), (\mathcal{T}_2, \tilde{\mathbf{x}}_9), (\mathcal{T}_1, \tilde{\mathbf{x}}_2), (\mathcal{T}_9, \tilde{\mathbf{x}}_4), (\mathcal{T}_3, \tilde{\mathbf{x}}_5)).$$

Finally, this remaining sequence of nodes is inserted into the blank spaces of c_i, starting at the second cut point, to obtain

$$c_i = ((\mathcal{T}_9, \tilde{\mathbf{x}}_4), (\mathcal{T}_3, \tilde{\mathbf{x}}_5) \mid (\mathcal{T}_5, \mathbf{x}_3), (\mathcal{T}_7, \mathbf{x}_4), (\mathcal{T}_4, \mathbf{x}_5), (\mathcal{T}_6, \mathbf{x}_6) \mid (\mathcal{T}_8, \tilde{\mathbf{x}}_7), (\mathcal{T}_2, \tilde{\mathbf{x}}_9), (\mathcal{T}_1, \tilde{\mathbf{x}}_2)).$$

Similarly, the second offspring is

$$c_{i+1} = ((\mathcal{T}_4, \mathbf{x}_5), (\mathcal{T}_6, \mathbf{x}_6) \mid (\mathcal{T}_7, \tilde{\mathbf{x}}_3), (\mathcal{T}_9, \tilde{\mathbf{x}}_4), (\mathcal{T}_3, \tilde{\mathbf{x}}_5), (\mathcal{T}_5, \tilde{\mathbf{x}}_6) \mid (\mathcal{T}_8, \mathbf{x}_8), (\mathcal{T}_1, \mathbf{x}_2), (\mathcal{T}_2, \mathbf{x}_1)).$$

III.B. Mutation

Mutation, in lines 8-11 of the pseudocode Tab. 1, is used in case the initial population is not rich enough to find a good solution via crossover alone. We use four different kinds of mutation. In *orientation shift*, an index $i \in \{1, \ldots, n\}$ is chosen uniform randomly, then the aircraft azimuth ψ within the state $\mathbf{x}_i$ is reset uniform randomly in the interval $[0, 2\pi)$ (radians). In *position shift*, an index $i \in \{1, \ldots, n\}$ is chosen uniform randomly, then the aircraft position (x, y) within the state $\mathbf{x}_i$ is reset uniform randomly within the polygonal approximation of $\mathcal{V}(\mathcal{T}_{k_i})$. In *swap*, two indices $i, j \in \{1, \ldots, n\}$ are chosen uniform randomly, then the nodes $(\mathcal{T}_{k_i}, \mathbf{x}_i)$ and $(\mathcal{T}_{k_j}, \mathbf{x}_j)$ swap positions within the chromosome. In *partial reverse*, two indices $i, j \in \{1, \ldots, n\}$ are chosen uniform randomly, then the segment of the chromosome $(\mathcal{T}_{k_i}, \mathbf{x}_i), \ldots, (\mathcal{T}_{k_j}, \mathbf{x}_j)$ is reversed; this includes rotating the azimuth portion of those nodes' states by π radians.

IV. Numerical Study

We have implemented our genetic algorithm for the PVDTSP in C++ on a 2.33 GHz i686. For comparison, a purely random search was conducted for the same amount of time that the genetic algorithm ran. The results from two Monte Carlo numerical studies are shown in Tab. 2, Fig. 3, and Fig. 4. In these porblem instances the aircraft minimum turn radius was $\rho_{\min} = 3$ m. The genetic algorithm produced tours on average nearly half the length of the pure random search.

V. Extension to Scenarios with Wind and Airspace Constraints

Recall that in Sec. II we formulated the minimum time reconnaissance path planning problem as finding a sequence of states $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ 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 genetic algorithm in order to evaluate the distance function $d(\mathbf{x}, \mathbf{x}')$ (which is all we need to evaluate chromosome fitness in the genetic algorithm). In this way, the minimum time state-to-state trajectory planner is a *module* within our genetic algorithm. We could therefore use our genetic algorithm with any of the minimum time state-to-state trajectory planners available in the literature. These include planners which can handle wind

Table 2. Statistics from examples computed by a C++ program on a 2.33 GHz i686.

Instance	No. of Runs	Computation Time per Run	Genetic Algorithm Parameters	Genetic Algorithm Mean Best Tour Length	Random Search Mean Best Tour Length
Fig. 3	50	2.55 s	$N = 50$, $N_g = 500$, $N_e = 4$ $p_c = 0.7$, $p_m = 0.1$	37.07 m	60.06 m
Fig. 4	50	10.86 s	$N = 100$, $N_g = 500$, $N_e = 4$ $p_c = 0.7$, $p_m = 0.1$	83.69 m	164.09 m

and no-fly zones. The literature on nonholonomic trajectory planning is vast, so we survey only briefly the most relevant.

Without obstacles or wind, the procedure for computing an optimal Dubins pose-to-pose[d] path was first shown using complicated measure theoretic arguments in,[3] then later Pontryagin's maximum principle from optimal control in[21] and.[22] More recently it has been shown that in a constant wind field without obstacles, a shortest pose-to-pose path with bounded turning rate can be caclulated by transforming the final pose to a so-called *virtual pose*, applying the no-wind method, then transforming back.[4] 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 v vertices, the shortest collision-free Euclidean path (no curvature constraint) can be calculated in $\mathcal{O}(v \log v)$ time.[23] Unfortunately the same problem with a Dubins vehicle is NP-hard in v.[24] However, much work has been done to quickly find nearly optimal obstacle avoiding paths, surveyed further in Ref. 25, 26. Trajectory planners specifically intended for fixed-wing UAVs, which use a branch and bound technique ,are described in Ref. 27, 28. Another approach to nonholonomic motion planning with obstacles is to use a MILP (Mixed Integer Linear Program).[29,30]

VI. Conclusion

We have formulated the general aircraft visual reconnaissance problem for static ground targets in terrain and shown that, under simplifying assumptions, it can be reduced to a variant of the Traveling Salesman Probem which we call the PVDTSP (Polygon-Visiting Dubins Traveling Salesman Problem). We designed a genetic algorithm to solve the PVDTSP and validated it in a Monte Carlo numerical study for scenarios with $n = 5$ and $n = 10$ targets. For fixed computation time, the genetic algorithm produced reconnaissance tours on average nearly half the length (and thus can be flown in half the time) of those delivered by a random search. The modular design of the genetic algorithm allows it to easily be extended to handle more realistic assumptions such as wind and airspace constraints.

We are currently conducting more numerical studies to test the sensitivity of the genetic algorithm with respect to its parameters. For the future we hope to conduct more test using different state-to-state trajectory planning modules, in particular to handle wind and airspace constraints. Other avenues to explore include varying the genetic algorithm parameters dynamically during runtime as in simulated annealing, or applying a local optimizer to each generation.

References

[1]D. Gross, S. Rasmussen, P. Chandler, and G. Feitshans, "Cooperative operations in UrbaN TERrain (COUNTER)," in *Defense Transformation and Network-Centric Systems*, vol. 6249 of *Proc. of SPIE*, pp. 62490G–1–11, SPIE, 2006.

[2]N. Jodeh, M. Mears, and D. Gross, "An overview of the Cooperative Operations in UrbaN TERrain (counter) program," in *AIAA Conf. on Guidance, Navigation and Control*, (Honolulu, Hawaii), 2008. Electronic Proceedings.

[3]L. E. Dubins, "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, pp. 497–516, 1957.

[4]T. G. McGee, S. Spry, and J. K. Hedrick, "Optimal path planning in a constant wind with a bounded turning rate," in *AIAA Conf. on Guidance, Navigation and Control*, (San Francisco, CA), Aug. 2005. Electronic Proceedings.

[5]J. Gross and J. Yellen, *Handbook of Graph Theory*. CRC, 1 ed., December 2003.

[6]G. Gutin and A. P. Punnen, *The Traveling Salesman Problem and Its Variations*. Springer, 1 ed., May 2007.

[d]By *pose* is intended position together with orientation, i.e., the state of a constant velocity Dubins vehicle.

[7]J. L. Ny, E. Frazzoli, and E. Feron, "The curvature-constrained traveling salesman problem for high point densities," in *IEEE Conf. on Decision and Control*, pp. 5985–5990, 2007.
[8]K. Savla, E. Frazzoli, and F. Bullo, "Traveling Salesperson Problems for the Dubins vehicle," *IEEE Transactions on Automatic Control*, vol. 53, no. 6, pp. 1378–1391, 2008.
[9]J. Le Ny and E. Feron, "An approximation algorithm for the curvature-constrained traveling salesman problem," in *Allerton Conference on Communications, Control and Computing*, (Monticello, IL), Sept. 2005.
[10]Z. Michalewicz and D. B. Fogel, *How to Solve It: Modern Heuristics*. Springer, December 2004.
[11]B. Freisleben and P. Merz, "A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems," in *In Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pp. 616–621, IEEE Press, 1996.
[12]P. L. naga, C. M. H. Kuijpers, R. H. Murga, I. Inza, and S. Dizdarevic, "Genetic algorithms for the travelling salesman problem: A review of representations and operators," *Artificial Intelligence Review*, vol. 13, pp. 129–170, April 1999.
[13]L. V. Snyder and M. S. Daskin, "A random-key genetic algorithm for the generalized traveling salesman problem," *European Journal of Operational research*, vol. 174, 2006.
[14]Z. C. Huang, X. L. Hu, and S. D. Chen, "Dynamic traveling salesman problem based on evolutionary computation," in *Proceedings of the 2001 Congress on Evolutionary Computation*, vol. 2, pp. 1283–1288, 2001.
[15]W. Pullan, "Adapting the genetic algorithm to the travelling salesman problem," in *Proceedings of the 2003 Congress on Evolutionary Computation*, vol. 2, pp. 1029–1035, 2003.
[16]V. Shaferman and T. Shima, "Cooperative uav tracking under urban occlusions and airspace limitations," in *AIAA Conf. on Guidance, Navigation and Control*, (Honolulu, Hawaii), Aug 2008. Electronic Proceedings.
[17]V. Shaferman and T. Shima, "Co-evolution genetic algorithm for UAV distributed tracking in urban environments," in *ASME Conference on Engineering Systems Design and Analysis*, Jul 2008.
[18]J. E. Bresenham, "Algorithm for computer control of a digital plotter," in *Seminal Graphics: Pioneering Efforts that Shaped the Field*, (New York, NY, USA), pp. 1–6, Association for Computing Machinery, 1998.
[19]S. K. Ghosh, *Visibility Algorithms in the Plane*. Cambridge University Press, 2007.
[20]J. C. Spall, *Introduction to Stochastic Search and Optimization*. Wiley-Interscience, April 2003.
[21]J.-D. Boissonnat, A. Cérézo, and J. Leblond, "Shortest paths of bounded curvature in the plane," *Journal of Intelligent and Robotic Systems*, vol. 11, pp. 5–20, 1994.
[22]X. N. Bui, J. D. Boissonnat, P. Soueres, and J. P. Laumond, "Shortest path synthesis for dubins non-holonomic robot," in *IEEE Transactions on Robotics and Automation*, 1994.
[23]J. Hershberger and S. Suri, "An optimal algorithm for euclidean shortest paths in the plane," *SIAM Journal on Computing*, vol. 28, pp. 2215–2256, 1999.
[24]J. Reif and H. Wang, "The complexity of the two dimensional curvature-constrained shortest-path problem," in *In Proc. Third International Workshop on the Algorithmic Foundations of Robotics*, pp. 49–57, 1998.
[25]J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.
[26]S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.
[27]A. Eele and A. Richards, "Path-planning with avoidance using nonlinear branch-and-bound optimisation," in *AIAA Conf. on Guidance, Navigation and Control*, (Hilton Head, South Carolina), 2007. Electronic Proceedings.
[28]A. Eele and A. Richards, "Comparison of branching strategies for path-planning with avoidance using nonlinear branch-and-bound," in *AIAA Conf. on Guidance, Navigation and Control*, (Honolulu, Hawaii), 2008. Electronic Proceedings.
[29]F. Borrelli, D. Subramanian, A. U. Raghunathan, and L. T. Biegler, "MILP and NLP techniques for centralized trajectory planning of multiple unmanned air vehicles," in *American Control Conference*, 2006.
[30]A. Richards and J. P. How, "Aircraft trajectory planning with collision avoidance using mixed integer linear programming," in *American Control Conference*, pp. 1936–1941, 2002.

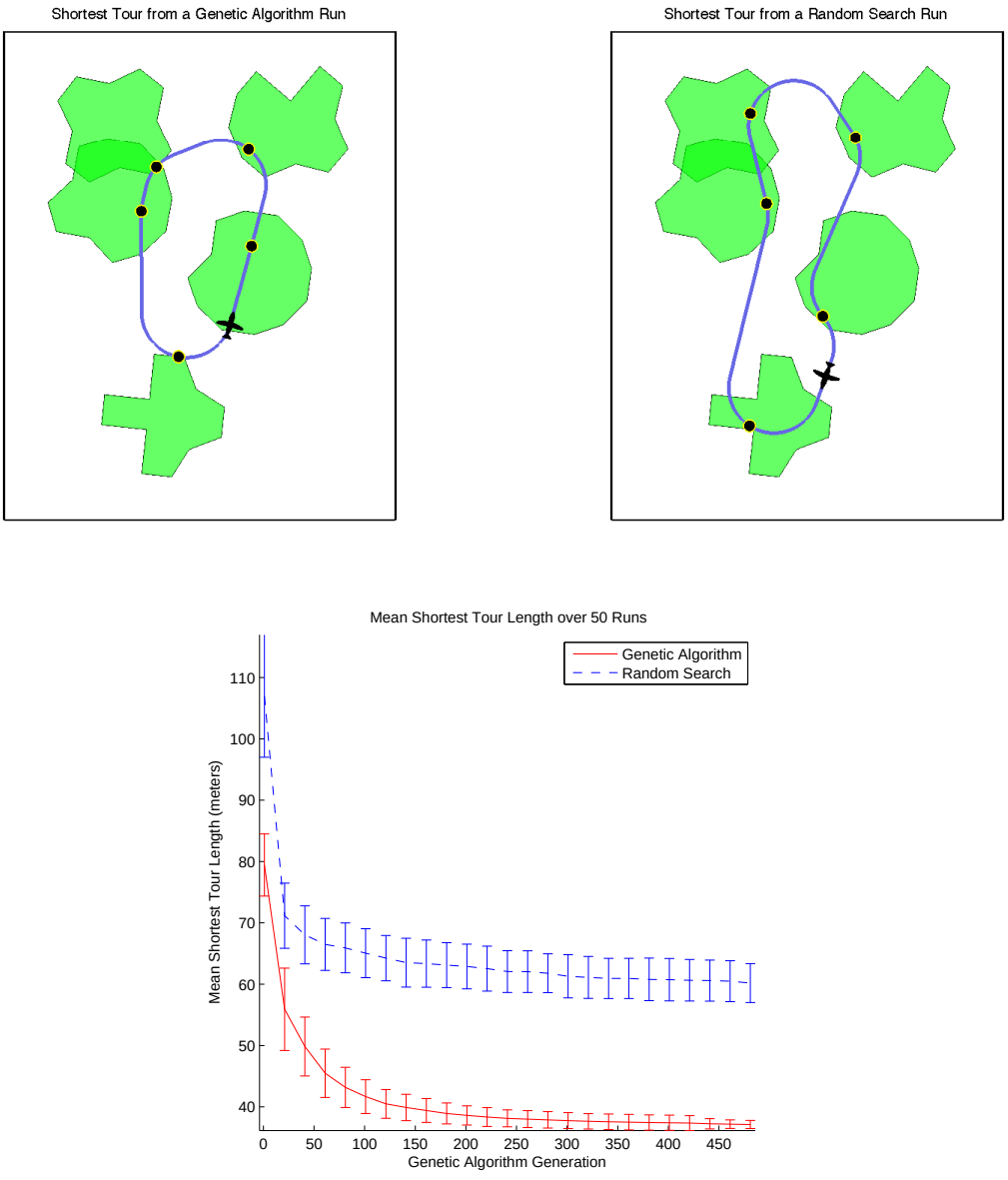
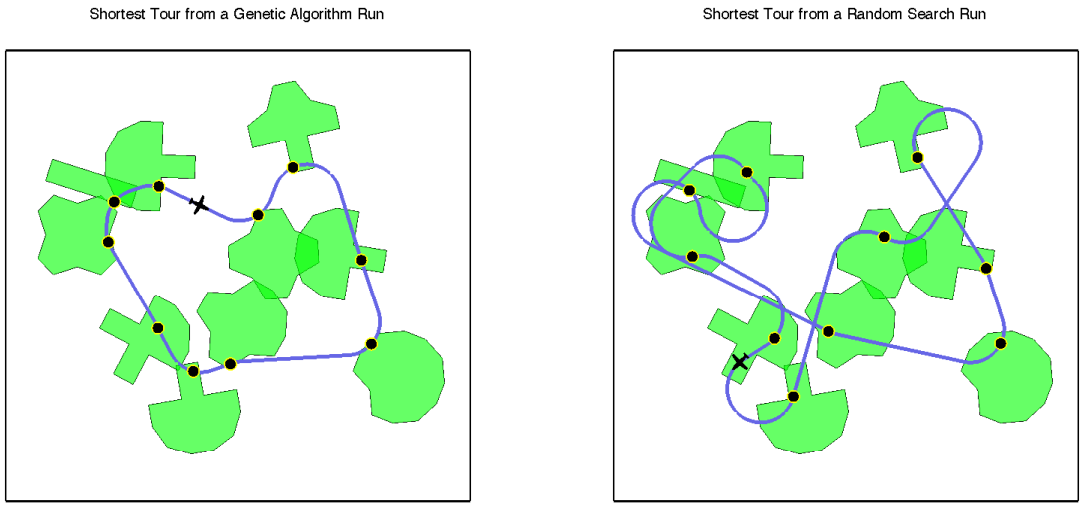


Figure 3. Computed example with $n = 5$ targets, aircraft minimum turn radius $\rho_{\min} = 3$ m. Green polygons represent the target visibility regions. Black dots are the tour nodes. Error bars show the sample standard deviations. Cf Tab. 2.



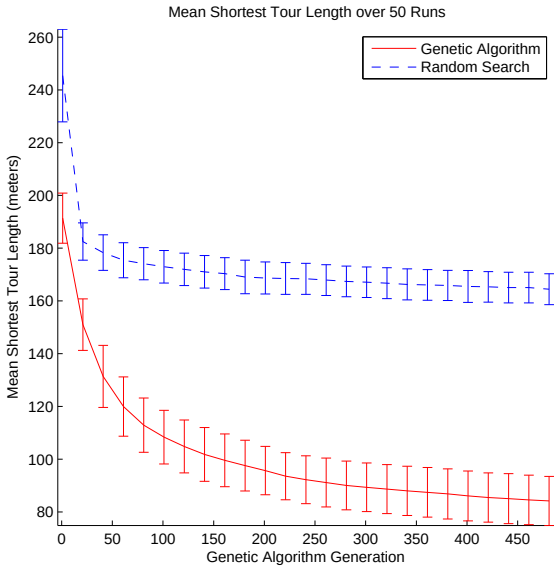


Figure 4. Computed example with $n = 10$ targets, aircraft minimum turn radius $\rho_{\min} = 3$ m. Green polygons represent target visibility regions. Black dots are the tour nodes. Error bars show the sample standard deviations. Cf Tab. 2.