Efficient Aerial Coverage Search in Road Networks

Michael Dille and Sanjiv Singh
The Robotics Institute, Carnegie Mellon University *

Wide-area coverage is well suited for small unmanned aircraft, however common coverage algorithms are particularly inefficient in many common environments such as urban areas. Classical open-loop spiral or lawn-mower patterns typically cannot represent large sub-regions that are unlikely to contain an object of interest that are thus wastefully covered. More general algorithms can maintain arbitrary likelihood distributions but fundamentally solve a complex continuous-space trajectory optimization problem, requiring a trade-off between look-ahead and computational complexity while providing at best asymptotic coverage guarantees. This paper considers generating global coverage patterns in the space of the coverage area, with specific focus on road networks, and mapping these to UAV actions. A new method is proposed for producing and following such patterns with dynamics-constrained vehicles, compared with existing coverage strategies, and shown to be advantageous in many realistic environments through simulations of real-world aircraft. Results suggest environment density as a metric for algorithm selection given a lack of dominance by any one strategy across all environments.

I. Introduction

Small field-launched unmanned aerial vehicles (UAVs) are particularly well suited to wide-area coverage tasks required in surveillance, mapping, and rescue applications. Such UAVs move faster and are less vulnerable to obstacles than ground vehicles, while providing less costly and more rapid deployment than larger aircraft requiring dedicated infrastructure and careful mission risk assessment. These benefits over larger aircraft come at the cost, however, of reduced payload capacity and maneuverability. Typically, small UAVs operate at relatively low altitude, with moderate speed, and carrying only limited sensors such as cameras, often fixed with respect to the air-frame.

A coverage task, therefore, becomes one of producing a path along which to steer the vehicle–and hence the relatively small footprint of an onboard camera–that results in all locations of interest passing under the sensor footprint at least once during the motion. This differs from most planning tasks in that the objective is not to reach an explicit goal location in minimum time, but rather to generate a path of minimal length providing coverage. Example applications include mapping, landmark detection, and search for stationary (in contrast to evasive) targets. Existing strategies commonly fall into one of two categories, both with substantial drawbacks. The first is to use pre-determined spiral or lawnmower-like patterns to exhaustively sweep an area, resulting in a very conservative search that may be very inefficient in sparse environments containing large areas known to not be of interest, yet without a straightforward means of incorporating such knowledge. The second is to plan deliberate motions that produce (ideally) a time-optimal coverage pattern given a prior likelihood distribution of area importance, which fundamentally represents a complex continuous-space trajectory optimization task that must be grossly approximated to maintain tractability.

This paper explores the alternative approach of planning coverage patterns in the abstract space of the area to be covered, with specific focus on the common case of road networks, which are then mapped back to a sequence of high-level UAV actions. A new method for generating and following such patterns is proposed that retains effectiveness for severely motion-constrained vehicles with high-level control interfaces that may be treated as a minimal model easily implemented by more capable aircraft. This, along with several variations, is compared in realistic simulation to existing open-area and road coverage strategies and shown to have several desirable properties while providing better performance in many common scenarios.

*Pittsburgh, Pennsylvania, USA, 15213 (Email: mdille3@ri.cmu.edu)

It is finally concluded that no single strategy is appropriate for all environments but that the best may be chosen prior to execution by evaluating environment density and rapidly simulating the expected run-time of several algorithms.

II. Related Work

Simple geometric area coverage search, or equivalently search for a target with uniform prior, was an early application of UAVs and received much attention but little formal study given its perceived simplicity. Ablavsky and Snorrason[1] provide commonly cited geometric strategies for a UAV modeled as a Dubins car (stated formally in Sec. III) using straight-line sweeps to form lawnmower, Zamboni, and box-spiral patterns. Quigley et al.[2] propose approximately uniform coverage by directing a UAV to orbit a point whose location slowly spirals outward from the center of an area to search. Extension to multiple vehicles typically takes the form of region assignment to avoid redundant coverage and produce multiple single-vehicle search tasks. Enns et al.[3] suggest extending single-UAV sweeping to interleaved sweeps by multiple vehicles, addressing turning constraints. Beard and McLain[4] consider similar forms of sweeping while avoiding hazards and maintaining a minimum communication distance between vehicles should this be required. Maza and Ollero[5] propose polygonal partitioning of complex areas, each of which is assigned to a UAV executing any simpler strategy. An alternative circular decomposition is put forth by Girard et al.[6] Crucially, all such methods by design cover exactly the entirety of a bounding polygon or circle and, except via iterative decomposition, cannot represent internal areas that need not be searched.

Search strategies have also been proposed to make use of a prior distribution of area importance to capture phenomena such as a reported target location likelihood, irregular outer bounding area shape, or known uninteresting interior regions. Motions may then be chosen to prioritize likely areas and eliminate very low-likelihood regions. Commonly, a cellular or particle representation is chosen to maintain search progress, to which a motion planner is applied to maximize an objective such as short-term detection probability. Bourgault et al.[7] present a famous early example, in which one or more UAVs each greedily move along the local gradient of a probability grid yet produce emergent coordinated behavior. Polycarpou et al.[8] provide a more general multi-step planning framework. Tisdale et al.[9] implement and field-test a multi-UAV sequential allocation scheme using horizon planning to maximize the short-term probability of target detection using Bayesian updates of discrete distributions of either cells or particles. Geyer[10] describes a similar strategy using a highly-optimized observation-predicting planner accounting for occlusions. An example of a different class of strategy is that given by Mathew and Mezic,[11] recently applied to UAV search, in which a relatively large number of vehicles move in a scattered fashion so as to maximize asymptotic uniformity of coverage in area subsets of interest. All such methods necessarily make a trade-off between myopia and computational complexity, providing sub-optimal short-term paths and asymptotically (if at all) guaranteed coverage in order to remain tractable.

For road networks, a natural and intuitive representation is the underlying road connectivity graph. Directly solving a graph coverage task within this graph produces, conceptually, an optimal coverage path in the space of the road network that may provide vital insight for aerial coverage. Given that intersections (graph nodes) are of relatively infinitesimal size compared to road segments (edges) and may in practice simply be considered endpoints of road segments, the most direct phrasing of the abstract problem statement is that of producing a minimal covering path in edge space (simply, a minimum-length connected sequence of edges containing each edge at least once). This is known in the graph-theoretic literature as the *Chinese Postman Problem* (CPP), a generalization of finding Eulerian paths (requiring that each edge be traversed exactly once), which has been shown to be solvable in polynomial-time[12] for strictly-undirected (most physically realistic) or strictly-directed graphs and for which fast practical algorithms exist.[13] However, though edges may be assigned costs indicating for instance coverage traversal time, a CPP-produced edge sequence may correspond to a highly suboptimal coverage trajectory. Most importantly, the CPP framework cannot capture the true UAV motion cost of moving *between* edges (instead assuming this cost to be zero) that in actuality depends on the relative orientation of edge endpoints. Further, it artificially constrains aircraft to the connectivity of the graph as though it were a ground vehicle, failing to take advantage of cases in which skipping across unconnected space could reduce total coverage time. Similarly, the fact that the true field of view is a finite area rather than a sweeping point is ignored, providing no means to represent overlapping coverage of multiple points in the environment from one vantage location. Finally, the CPP task does not scale directly to multiple searchers, as all complexity results and fast exact algorithms of which the authors

are aware are stated for the single-vehicle case. Oh et al.[14] explore several ad hoc heuristics for extending CPP solutions to multiple vehicles, with limited success.

To address several of these limitations, the study of the CPP problem has previously been extended to several variants including the *modified CPP* (mCPP) that treats the graph edges as a basket of segments without artificially constraining their coverage to the graph's connectivity and the *modified Dubins CPP* (mDCPP) that further adds an inter-edge cost derived from the motion cost for a Dubins vehicle to move between them. Oh et al.[14] also explored phrasing the mDCPP as a multi-choice multi-dimensional knapsack problem using mixed integer linear programming, which fully represents the minimization of the longest Dubins path of any vehicle but is clearly intractable for any substantially-sized environment. Oh et al.[15] later presented an alternative approach to the mDCPP task that greedily builds up a coverage sequence by incrementally extending the current sequence, inserting each edge at the current-best point within the existing sequence while accounting for vehicle maneuvering costs between each edge. This suffers from several weaknesses including inevitably local optimality, potentially excessive maneuvering and hence further reduced optimality due to fixing the choice of sweep direction when an edge is added, runtime quadratic in the number of edges that is somewhat slow in practice, example simulations demonstrated only on unrealistically widely distributed edges not typical of any real road network, and no proof of performance guarantees for multiple vehicles. Nonetheless, implementation by the authors has shown this algorithm to work quite well in many scenarios, and it is therefore used as a primary point of comparison in this paper.

A related path-optimization problem in graphs is the well-known *Traveling Salesman Problem* (TSP), in which an agent wishes to visit each node in a graph while minimizing intermediate transit costs. It is typically phrased for a single agent on complete graphs, so the additional constraint of visiting each node exactly once is added without loss of generality assuming edge costs adhere to the triangle inequality. A similar problem is the *Vehicle Routing Problem* (VRP), a multi-vehicle generalization modeling delivery scheduling that is typically phrased so as to minimize total team travel cost while not exceeding a per-vehicle capacity corresponding to sums of per-node demand. Neither of these problems directly solves the coverage task of interest as they operate on nodes rather than edges and, as with classical edge-oriented problems, do not capture path-dependent vehicle motion cost. Further, both are NP-complete for even relatively restricted cases, with the TSP itself being a canonically hard problem to reduce from when demonstrating the NP-hardness of other problems. However, their natural form as representing the cost of paths through abstract states and their many physical instantiations have led to fast heuristics, with a common Lin-Kerninghan implementation[16] providing good solutions for thousands of nodes in several seconds. Recent work in the aerial domain for mission planning and point-of-interest visitation have studied the integration of true motion costs and the development of bounds on sub-optimality if they are neglected.[17–19] This paper proposes a coverage strategy building upon the TSP that maps edge coverage to a node-visitation problem that corresponds well to the inherent capabilities of small UAVs.

Other related and important problems that are beyond the scope of this work include the detection and tracking of objects of interest observed during coverage, accurately geolocating such objects using the available limited-accuracy sensors, and producing trajectories to maintain view or improve a geolocation estimate of an object deemed to be of interest. Existing strategies for each of these are summarized in a previous paper,[20] with field-tested tracking and geolocation methods implemented by the authors described in another.[21]

III. System Model

As a model for UAV motion, the well-known Dubins vehicle[22] is adopted. Though a gross simplification of UAV aerodynamics, it captures the fundamental constraints of fixed-wing aircraft including a minimum forward speed and turning radius and is commonly used in the small UAV search and tracking literature.[23–25] This planar kinematic model for state and its derivative is defined by

$$\mathbf{X} = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix} \text{ and } \dot{\mathbf{X}} = \begin{bmatrix} V\cos\theta \\ V\sin\theta \\ u \end{bmatrix}, \tag{1}$$

where $V > 0$ is the current forward speed of the aircraft and u is the steering input control constrained by a maximum turn rate defined by $|u| \leq \dot{\theta}_{max}$ that equivalently constrains a minimum turning radius

$R_{min} = V/\dot{\theta}_{max}$. For simplicity, V and altitude are assumed to be constant, reducing the problem to a physically two-dimensional single-input one, but may be allowed to vary with little consequence.

To further model small field-launched UAVs, it is also assumed that the aircraft is equipped with a high-level autopilot that can be given a command indicating either a sequence of waypoints to pass through or a loiter or orbit point that is then traced using for instance Lyapunov vector fields.[24] The control input may be formally stated as

$$\mathbf{u} = \begin{cases} [\mathbf{p}_0, \ldots, \mathbf{p}_n], \mathbf{p}_i \in \mathbb{R}^2 & : \text{waypoint mode} \\ \mathbf{o} \in \mathbb{R}^2 & : \text{orbit mode} \end{cases} \tag{2}$$

Rather than acting as a simplifying assumption, this is intended to provide generalization in that capable aircraft with low-level control accessibility can easily provide such an interface, whereas such an abstraction may be all that is exposed or readily implementable on simpler widely-available commercial UAVs. Naturally, the radius of an orbit must be at least the vehicle's minimum turning radius, but larger radii are possible if supported by the autopilot. For simplicity, the orbit radius will be assumed to be the minimum turning radius.

These motion and control models primarily describe fixed-wing aircraft, however they may be equally applied to most rotorcraft at the cost of not making use of additional capabilities such as hovering and instantaneous motion in arbitrary directions. Here again, the choice is made to provide a more generally applicable strategy. Indeed, in real-world use, fixed-wing aircraft are often preferable due to their higher speeds, greater mission durations, design simplicity, larger payload capacity, and improved robustness over rotorcraft.

Finally, the coverage sensor is assumed to be a fixed-zoom camera that may be attached to a positionable gimbal or fixed with respect to the aircraft body. To reduce planning complexity and maintain applicability to simple designs, only a statically-positioned camera is considered. Two classes of positions are chiefly of interest: side-aimed cameras and forward or directly-downward facing cameras, all angled towards the ground so as to minimize useless above-horizon views. The prior class is popular in surveillance applications because a single point of interest may be kept in constant view by orbiting it, whereas the latter must (within a motion model requiring forward velocity) overfly a point and return to it. A complementary property is vulnerability to occlusion from terrain obstacles, for which side-facing cameras may suffer from directional viewing, whereas a direct flyover will avoid all but overhead obstacles.

Though largely applicable to downward- and forward-facing cameras as well, the proposed strategy focuses on side-facing cameras owing to several convenient practicalities. Most importantly, while the UAV orbits a point with the camera pointed inward, a finite region (conservatively describable as a circle having radius approximately half the orbit radius for typical vehicle parameters described in Section VI) around the center of the orbit remains in view at all times, even taking into account vehicle roll aerodynamically required to produce the turn. For conciseness, the "radius of field of view" of this consistently viewed interior region is henceforth written R_{FOV}. The immediate implication is that to view an area around a given point, the aircraft needs only to momentarily lie anywhere on the orbit surrounding that point. This is depicted pictorially in Figure 1. More generally, an arbitrary viewing trajectory (covering a finite region of width typically equal to the orbit radius) may be followed, even one containing arbitrarily sharp turns: the trajectory may simply be followed with a lateral offset of one turning radius, and where the turning rate is exceeded, the UAV need only continue an orbit around the current location until aligned with the desired tangent vector.

IV. Basic Tessellated Node Coverage Strategy

The proposed strategy takes inspiration from the fact that under the models described, a UAV need only reach some point in an orbit around a location to cover a surrounding region. By judiciously placing and sequencing orbit centers so as to minimize their number and inter-orbit motion costs, efficient coverage may be obtained. Though designed to be appropriate for Dubins vehicles, it does not fully optimize the Dubins motion cost of the coverage path. Section V explores ways to improve upon this to better capture true motion costs.

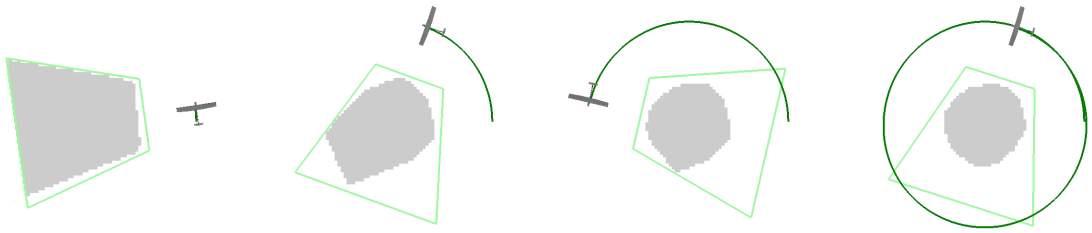


Figure 1: Depiction of the consistently viewable region while orbiting a point using a fixed, left-facing camera. The dark green trace indicates the UAV's trajectory, the light green trapezoid is its instantaneous field of view on the ground, and the gray region shows the area that has continuously been in view since the start of the orbit. As can be seen, at any point around an orbit, regardless of when the orbit was begun, a circularly-shaped area having radius approximately half the orbit radius will always be in view.

IV.A. Algorithm Description

At a high level, the strategy may be described as consisting of several simple steps.

1. **Tessellate regions to be covered with sensing-footprint-sized circles**

   The road graph $(\mathbf{I}, \mathbf{S})$ of intersections $\mathbf{I}$ and interconnecting segments $\mathbf{S}$ is discretized, with circles of radius R_{min} placed along each edge having centers spaced at most distance d apart producing a set $\mathbf{O}$. Given the prior observation that the intersection of sensor footprints around an orbit may be conservatively represented as a circle with half that of the orbit, $R_{FOV} = R_{min}/2$ is chosen as the value of d. Additional regions of interest not lying on roads may be added, for instance by overlapping tiling. This provides applicability to environments not strictly in the form of a road network, such as one containing small pockets including lots, parks, or fields. The resulting discretization represents a collection of orbits which, if the UAV enters each at least momentarily, provide full coverage of the environment.

   Where edges lie in close proximity (such as near intersections and more frequently in dense environments) and with possibly large values of R_{FOV} (more than 60m for the parameters considered in Section VI), this coverage discretization may contain substantial redundancy, in that many circles may be safely removed without sacrificing coverage completeness. Optimally choosing a minimal subset of these is a special case of the *Planar Sensor Cover Problem*, itself a very hard problem.[26] For the purposes of evaluation in this paper, simple greedy circle elimination is used, still achieving substantial reductions.

   Figure 2 provides an example reduced tessellation of a small road network representable as a graph.

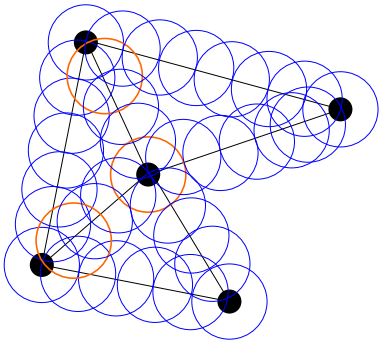


Figure 2: Example tessellation by orbits of a small environment, with easily-eliminated redundant orbits highlighted in orange. Larger environments and better tessellation provide more opportunities for redundant coverage elimination.

2. **Provide as visitation points to graph tour generator, producing an orbit sequence**

The problem is now one of choosing a sequence of and path between this set of orbits. To accomplish this, it is treated as an instance of a discrete routing problem (a TSP or VRP). A complete graph is synthesized, with nodes corresponding to orbits ($\mathbf{V} = \mathbf{O}$) and edges to the action of traveling between them ($\mathbf{E} = \mathbf{O} \times \mathbf{O}$). As the true cost of moving between orbits is path-dependent (upon which orbits immediately precede and succeed the current one), the substantial simplification is made to use the Euclidean distance between orbit centers as the edge cost. The impact of this on path optimality is studied in the next subsection, with mitigating improvements then suggested.

For a single vehicle, this directly corresponds to a traditional TSP, and with the use of a complete graph and Euclidean distance metric, the use of existing heuristic solvers highly-optimized for this common scenario is possible. For multiple vehicles, the problem is technically a *mini-max multiple TSP*, since it is most likely desired that the length of the longest path among all vehicles be minimized so as to optimize for overall mission time (rather than, for instance, total distance traveled or roughly equivalently, total flight time or energy expended). This problem is substantially less well studied, and though more elaborate solutions may be applied, repeated invocations of a capacitated VRP were used to approximately search for a multi-vehicle solution for the purposes of comparison in this paper. VRP problems are also typically phrased to assume a single "depot" location (a visitation point shared by all vehicles), which may be taken to be either a common launch location (if all are launched from one place) or an artificially chosen location whose distance to all initial vehicle locations is minimized.

3. **Choose the nearest point on the tour for each vehicle and tour direction**

   The output of a routing problem solver is a closed sequence of orbits to pass between for each vehicle. This tour does not necessarily pass through the initial location of each vehicle (though it may be explicitly directed to do so, for instance by adding an additional visitation point), and so a point on the tour must be selected. For simplicity, the orbit having lowest Dubins path cost to reach is selected. Likewise, the expected distance to travel if the tour were followed in the forward and reverse directions (these may differ due to the path following scheme next described) are compared, and the lowest-cost direction is selected.

4. **Follow the orbit sequence with each vehicle in parallel**

   For UAVs equipped with a side-looking camera, once on the tour, orbits are passed through in the chosen sequence by maintaining the current orbit until the vehicle's heading vector aligns with the tangent vector between the current orbit center and the next one. Once aligned, a straight-line path is taken to the next orbit, and since all orbits are of the same radius, the vehicle will eventually lie on the next orbit, and the procedure may be repeated. Alignment followed by straight-line motion will always be possible because this represents an instance of either the LSL or RSR Dubins motion case (consisting of an orbit plus a motion between orbit outer tangents).

   For UAVs equipped with a forward- or downward-looking camera, any of several published strategies to best follow TSP tours with Dubins vehicles may be invoked. Most naively, the locally optimal Dubins path from the current pose to the next orbit center may be followed, with a terminal orientation that must be chosen, either by search or by arbitrarily fixing it to the direction towards the succeeding orbit center, for instance. A formalization of the latter is the "alternating algorithm" proposed and analyzed by Savla et al.,[17] who also proposes a "bead-tiling" algorithm more appropriate for very dense collections of points.

5. **Return vehicles to their start locations**

   Being a closed path, the tour of orbits for each vehicle will terminate at its initial orbit, which by design either was or is close to its starting (e.g. take-off) location. Thus, an optimized overall mission plan is provided, leading from launch to recovery.

IV.B. Analysis

Naturally, one wishes to evaluate the performance of this strategy, particularly in the sense of overall mission duration optimality. From the start, this is not necessarily optimal among all possible coverage strategies, as the choice to tessellate with orbit-sized circles was made as part of a larger effort to trade off optimality for feasibility and is not easily evaluated. However, the optimality of the path cost through the sequence of

orbits (and hence mission duration) may be studied, with overall strategy performance left to experimental comparison. The fundamental source of suboptimality is the use of Euclidean distance between orbit centers as an approximation of the motion cost for a Dubins vehicle to move between orbits.

For the special case of side-facing cameras forming the focus here and the aforementioned path following strategy of maintaining an orbit until aligned with the next one in the sequence, some simple analysis may be applied to determine the amount of path length inflation over a hypothetical vehicle having no motion constraints. Specifically, we consider the worst-case additional distance that must be traveled while moving between orbits over the cost of moving directly between orbit centers. For a given sequence, an upper bound of π radians per orbit on average may be easily established because if the average were higher, simply following the tour in the opposite direction will reduce this below π (in general, if an orbit is entered at absolute heading θ_1 and left at absolute heading θ_2, traversing $\Delta\theta$ in the interim, then entering the orbit at θ_2 and leaving at θ_1 will traverse $2\pi - \Delta\theta$). Meanwhile, a lower bound on a worst case average may also be seen to π radians through the construction of an adversarial environment. Consider an environment consisting of only parallel segments of length R_{FOV} forming a set of stairs. Regardless of direction, every other corner requires maintaining an orbit for either $\frac{\pi}{2}$ or $\frac{3\pi}{2}$ to align with the next segment, for an average of π for a sufficiently long sequence. Thus, this bound is tight, and the additional maneuvering required may reach $\pi R_{min}|\mathbf{O}|$. This is substantial and undesirable, both because it can be quite large and because it grows linearly with the size of the environment. However, it is claimed that environments containing highly excessive maneuvering are contrived rather than typical as well be shown in simulation, and the following section suggests several extensions to improve upon this.

For forward- and downward-looking cameras, for which orbit visitation implies momentary visitation by the UAV itself (possibly with some posterior standoff), existing path following strategies and accompanying optimality bounds may be considered. For instance, Savla et al.[17] present an inflation bound for the the "alternating algorithm" given by an increase over the Euclidean path length of at most $\tau\pi R_{min}$ per alternate pair of visitation points (so that the total path cost is approximately $\frac{N}{2}\tau\pi R_{min}$) where $\tau \in [2.657, 2.658]$ and N is the number of points. Note that this exceeds πR_{min} on average, and hence that the inflation provided bound for side-facing cameras is lower.

V. Coverage Strategy Extensions

To address the potentially substantial path suboptimality shortcomings of the basic proposed method, several extensions are considered that better represent the true Dubins motion cost of moving between coverage visitation locations. For the first two, the same orbit-based tessellation and path following is considered, but the synthetic graph fed to the abstract tour generator is varied to produce more physically optimal sequences. The third operates directly on the set of road segments, synthesizing a graph based upon it.

V.A. Exact (Pairwise) Motion Cost Encoding

Due to the path-dependent nature of true motion costs, a graph consisting of merely positions is insufficient to capture these costs. Instead, a hypergraph may be constructed in which each vertex represents the state of being at an orbit having come from a prior orbit and edges represent the exact total Dubins cost of (a) maintaining the current orbit from an initial orientation aligned with the prior orbit to an alignment with the next orbit followed by (b) the straight-line motion to the next orbit. Formally this may be stated as

$$\begin{aligned} \mathbf{V} &= \mathbf{O} \times \mathbf{O} \\ \mathbf{E} &= \{(O_i, O_j) \rightarrow (O_j, O_k) | O_i, O_j, O_k \in \mathbf{O}\}, \end{aligned} \tag{3}$$

noting that this is no longer a complete graph and that edge costs are given by

$$cost\left((O_i, O_j) \rightarrow (O_j, O_k)\right) = 2\pi \cdot angle\left(\frac{O_j - O_i}{|O_j - O_i|}, \frac{O_k - O_j}{|O_k - O_j|}\right) + |O_k - O_j|, \tag{4}$$

where $angle(v_i, v_j)$ represents the angle between two vectors v_i and v_j. This construction is depicted graphically in Figure 3.

A graph of this form presents several complications. The first is that an orbit sequence may not be derived by simply applying a TSP solver, as this will return a tour containing *all* nodes, revisiting each node $|O|$

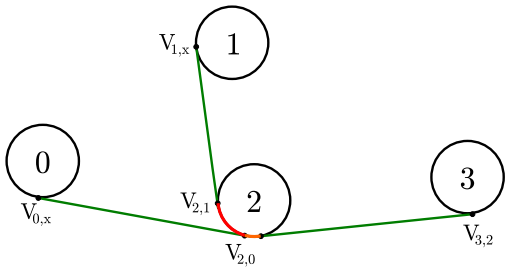


Figure 3: Graphical representation of exact (pairwise) motion cost encoding. Note the path segments highlighted in orange and red. The cost of moving from node 0 to 3 via 2 includes the maneuver around node 2 in orange, and the cost of starting at node 1 instead further includes the additional maneuver in red. In contrast, using only the Euclidean distances between orbit centers as inter-node cost neglects both, capturing only the green segments.

times. Instead, this may be seen as a *Generalized TSP* or a discrete instance of a *TSP with neighborhoods*, since nodes may be separated into $|O|$ disjoint clusters or neighborhoods, defined by

$$\mathbf{C_i} = \{(O_j, O_i) | O_j \in \mathbf{O}\}, \tag{5}$$

providing a separation into clusters each representing the physical state of presence at a given orbit, grouping all possible predecessor orbits.

Using a similar strategy employed by Isaacs et al.,[27] this may be converted from this form of TSP that is not easily solved to a more common *asymmetric TSP* (ATSP) using the Noon and Bean transform[28] without any growth in the graph. Finally, an ATSP may be further transformed as described by Kumar and Li[29] (building upon the idea incompletely proposed by Jonker and Volgenant[30]) to a symmetric TSP, at the cost of doubling the number of nodes so that fast heuristic symmetric solvers may be applied.

A second complication is that a graph on $2|\mathbf{O}|^2$ nodes is much larger than the original graph, which itself may be large already as a tessellation of a vast environment. Indeed, for sufficiently large coverage area, this may prove intractable. Where this is the case, the approximation proposed as the next strategy extension may prove adequate. However, sparse environments containing up to one hundred nodes produce graphs on less than ten thousand nodes that require tractably many edge cost evaluations and are still readily solved by modern TSP heuristics.

It should further be noted that using a heuristic symmetric TSP solver (returning inexact, suboptimal solutions) as the foundation may result in node sequences that are not technically valid solutions to the full problem with the aforementioned transformations applied (such as including node re-visitation). However, for the purposes of this application, this may be considered simply further weakening of the heuristic output and may be evaluated empirically. In practice, it was noted that invalid solutions were returned extremely rarely, and within several repeated iterations using differing random seeds for the heuristic solver, valid solutions were always found.

V.B. Approximate Quantized-Pose (Angle-discretized) Motion Cost Encoding

Where quadratic growth in the graph provided to the TSP solver is impractical, an alternative may be used as an approximation. For some chosen number of samples n, equally spaced points around each orbit are selected, forming a neighborhood of locations, one of which for each orbit must be visited to achieve total coverage. A complete graph is then formed, with edge costs defined by the cost of following the first orbit starting at the initial pose until aligned with the second orbit, moving onto the second orbit, and then

following the second orbit until the terminal sampled pose is reached. Formally, this may be stated as

$$\begin{aligned}
\mathbf{V} &= \mathbf{O} \times \{1, ..., n\} = \{V_{i,j} | i = 1..|\mathbf{O}|, j = 1..n\} \\
\mathbf{E} &= \mathbf{V} \times \mathbf{V} \\
\theta_j &= \tfrac{2(j-1)}{n}\pi \\
loc(V_{i,j}) &= loc(O_i) + R_{min} < \cos\theta_j, \sin\theta_j > \\
cost(V_{i,j} \to V_{k,l}) &= 2\pi \cdot angle\left(< \cos\theta_j, \sin\theta_j >, \tfrac{O_k - O_i}{|O_k - O_i|} \right) \\
&\quad + |O_k - O_i| \\
&\quad + 2\pi \cdot angle\left(\tfrac{O_k - O_i}{|O_k - O_i|}, < \cos\theta_l, \sin\theta_l > \right),
\end{aligned} \tag{6}$$

where ϕ is $\frac{\pi}{2}$ for left-facing cameras, 0 for forward-facing cameras, etc. and counterclockwise-positive heading angles are assumed (as for instance arise in a North-East-Down coordinate convention).

As with the prior hypergraph formulation, a GTSP may be formed by clustering vertices corresponding to the same orbit, defined formally as

$$\mathbf{C_i} = \{V_{i,j} | j = 1..n\}, \tag{7}$$

and solved by the same means. This produces a graph with $n|\mathbf{O}|$ vertices, with n selectable based on the desired resolution, providing a means to variably trade off between optimality and computational complexity. This formulation is shown in Figure 4.

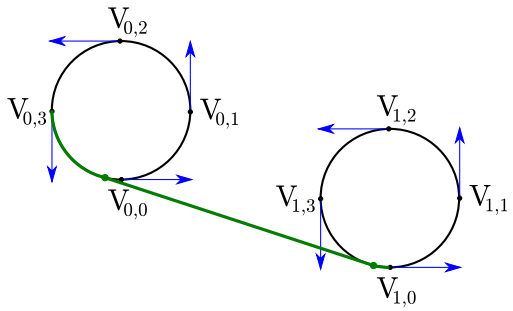


Figure 4: Graphical representation of approximate quantized-pose motion cost encoding with $n = 4$. In general the phase angle of pose samples is a free variable, here chosen so that poses are axis-aligned. Blue arrows indicate locations and corresponding orientations sampled around each orbit, and the green path shows the resulting path between two sample poses, comprised of a straight-line tangent segment having the same length as the Euclidean distance between orbit centers and two maneuvers for each orbit linking this segment to the desired quantized poses.

V.C. Cost-aware Road Segment Sequencing

Taking inspiration from both the mDCPP algorithm of Oh et al. and clustered graph formulations on which a GTSP is solved, an alternative graph may be synthesized directly from the set of road segments $\mathbf{S}$. In this variant, two nodes are assigned to each segment, one corresponding to the pose at the start of the edge along each direction. A complete $|\mathbf{S}|$-partite graph is then formed by connecting each node to all nodes except its opposite-direction partner, with edge costs defined by the Dubins motion cost between poses. Formally, this may be stated as

$$\begin{aligned}
\mathbf{V} &= \mathbf{S} \times \{0, 1\} = \{V_{i,j} | i = 1..|\mathbf{S}|, j = 0..1\} \\
\mathbf{E} &= \mathbf{V} \times \mathbf{V} \\
[P_{i,0}, \theta_{i,0}] &= [S_{i_0}, atan(|S_{i_1} - S_{i_0}|_x, |S_{i_1} - S_{i_0}|_y)] \\
[P_{i,1}, \theta_{i,1}] &= [S_{i_1}, atan(|S_{i_0} - S_{i_1}|_x, |S_{i_0} - S_{i_1}|_y)] \\
cost\left((V_i, j) \to (V_k, l)\right) &= |S_{i_0} - S_{i_1}| + dubins_cost([P_{i,(1-j)}, \theta_{i,j}], [P_{k,l}, \theta_{k,l}]),
\end{aligned} \tag{8}$$

where $(S_{i_0}, S_{i_1}) \in \mathbf{S}$ and $dubins\_cost([(x_1, y_1), \psi_1], [(x_2, y_2), \psi_2])$ represents the motion cost of the optimal Dubins motion between two planar poses.

Nodes clusters are then defined by

$$\mathbf{C_i} = \{V_{i,0}, V_{i,1}\}, \tag{9}$$

producing a graph with $2|\mathbf{S}|$ nodes, with each cluster containing two nodes. This is intuitively presented in Figure 5.

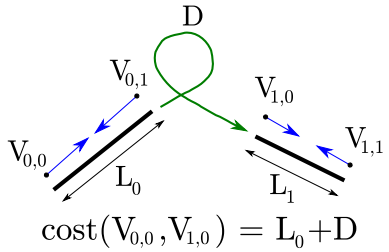


Figure 5: Graphical representation of cost-aware road segment sequence encoding. Each edge is assigned two nodes in the synthetic graph, corresponding to being located at opposite endpoints with pose pointed inward (as though about to sweep that edge). The motion cost between any two synthetic nodes is the cost of sweeping the edge the first node lies on plus the cost of a minimal maneuver between the endpoint pose of that sweep and the pose of the second node.

Solving the GTSP problem on this graph produces a minimal-length path that sweeps all road segments, fully taking into account inter-segment motion costs, encoded as a sequence of segments and a direction in which to sweep. This directly and globally solves the mDCPP problem, without a need to approximate motion costs or resort to a greedy solution, and it is highly tractable since a GTSP on up to several hundred road segments may be solved quickly with even exact TSP solvers. This method is henceforth referred to as the sweeping generalized TSP (SGTSP).

This method does have several caveats. First, as phrased, road segments are assumed to be linear, however this may be easily extended to arbitrary UAV-sweepable trajectories whose arclength is known. More importantly, tessellation of any open areas must be done with short segments rather than by placing orbits, and redundant coverage is not easily eliminated as coverage of only parts of road segments may overlap. Finally, since physical vehicles cannot actually execute tight Dubins maneuvers containing discontinuous curvature, the resulting trajectory may not be feasible, whereas the other methods built upon orbits are more conservative.

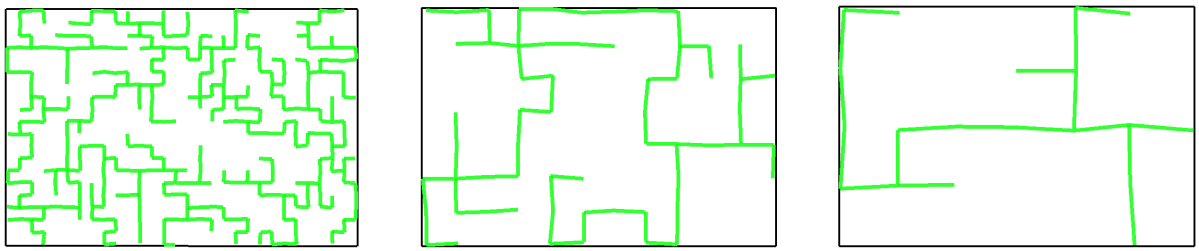
VI. Experimental Results

The proposed method and its extensions were evaluated in extensive realistic simulation to both demonstrate its efficacy and explore which algorithms are best suited to what forms of environments. Simulations were performed using a fixed-wing aircraft model very roughly modeled on the AeroVironment Raven,[31] the most widely deployed small field-reconnaissance UAV. As such and to emulate commercial autopilots with limited control authority, a fixed speed of 15m/s, a fixed altitude of 100m, and a turning radius of R_{min} = 125m were assumed, implying a sensor spot size of R_{FOV} = 62.5m. Within the simulation, algorithms were presented with a control interface described by Equation 2, however internally a high-fidelity kinematic model including a coordinated turn constraint inducing roll during turns was used. Camera parameters were also chosen for physical realism, with an analog NTSC-like resolution of 640x480 pixels, a horizontal field of view of 42 degrees, and a vertical field of view of 28 degrees. As side-facing cameras are the focus herein, a fixed left-pointing camera aimed downward 30 degrees below the horizon was chosen. For the purposes of visualization and evaluation only, a cellular Bayesian estimator similar to that proposed by Tisdale et al.[9] was used, with road segments finely discretized and each cell containing the probability of a target remaining assuming initially total contamination ($p = 1.0$ for all cells).

Three methods are initially primarily compared. The first is the aforementioned basic tessellated node coverage strategy, abbreviated OTSP. This is evaluated against a standard lawnmower-like Zamboni coverage pattern, which interleaves alternating axis-aligned sweeps of the entire length of an environment so as to avoid turns exceeding R_{min}. The third is the previously described mDCPP algorithm proposed by Oh et al.,[15] which may be summarized as iteratively growing a sequence of road edge sweeps by greedily adding

the edge whose insertion into the sequence (taking into account inter-edge Dubins maneuvers) increases the path cost by the least, with deconfliction between multiple vehicles provided by an auction-based scheme assigning the conflicted edge to the vehicle whose path cost would worsen the most if it did not receive that edge.

Comparative simulations were performed over a spectrum of randomly-generated environments intended to realistically mimic physical road networks. These were generated by starting with a dense 4km $\times$ 6km grid network with specifiable block size (edge length), jittering intersection locations, and removing a specifiable fraction of edges. The combination of edge length and fraction of edges removed conceptually correspond to environment density. In these simulations, 30% of edges were retained, and edge lengths were varied between 100m and 1km, corresponding to a range encompassing dense urban, suburban, and sparse rural. Example environments are shown in Figure 6. Very roughly, for the purposes of comparison here, environments with edge lengths within the range 100-300m will be referred to as urban, 300-700m as suburban, and 700m-1km as rural.

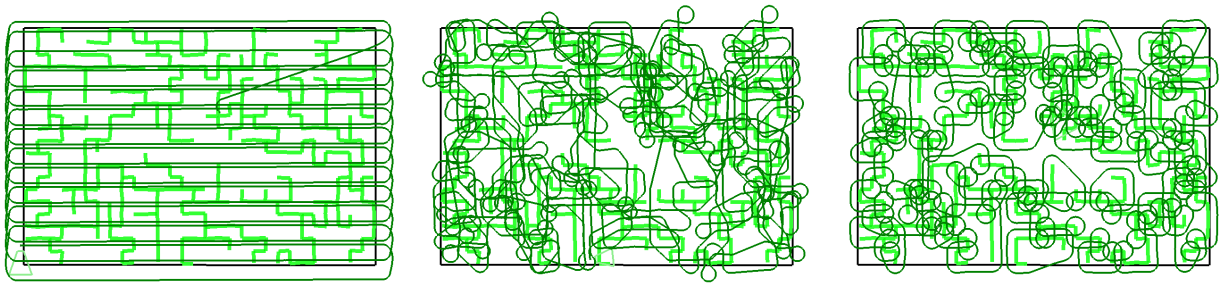


(a) Urban (b) Suburban (c) Rural

Figure 6: Sample environments along the density spectrum

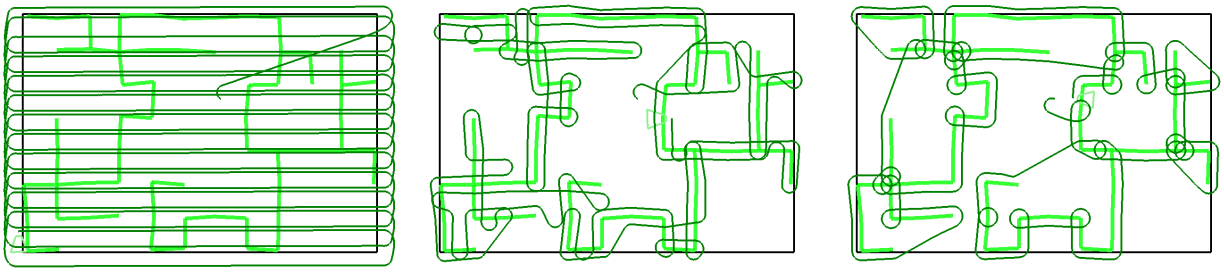
Several hundred simulations for each of 10 levels of environment density were performed, each with differently randomized environments and vehicle starting poses. For each run, the algorithm precomputation time, time to mission completion, and mission success rate (whether total coverage was achieved) were recorded. By design, total coverage was achieved in all cases. A typical run highlighting the strengths and weaknesses of each method over urban, suburban, and rural environments is shown in Figures 7, 8, and 9 respectively.

Results for a single vehicle are summarized in Figure 10. From this it may be seen that for dense environments (having average road segment length less than approximately 250m), Zamboni patterns are 2-5 times faster than either OTSP or mDCPP coverage. However, for less dense environments, this is inverted, with OTSP and mDCPP performing up to about 4 times as quickly. For a large density range roughly spanning sparse urban through suburban, OTSP is faster than both Zamboni and mDCPP, with a peak improvement of approximately 20% over mDCPP (at which point it is also $\frac{1}{3}$ faster than Zamboni). This is largely due to the elimination of redundant coverage in dense areas by the removal of some orbit tessellations, while in sparser environments many orbits are still required. For suburban through rural densities, SGTP is superior to all others, by up to 10%, while matching the performance of mDCPP for denser environments (indeed, any loss to mDCPP is attributable to experimental error and the use of a heuristic TSP solver). As this can be seen as a global version of mDCPP, this highlights the good performance of greedy assignment within mDCPP in areas having much to cover in all directions (urban) and the consequences of poor assignment for longer sweeps (rural). The pairwise-cost extension is omitted as it proved intractable for all but rural environments, and the quantized-pose extension is not shown as it performs poorly owing to a large problem space and sparse connectivity between synthetic nodes, both properties that reduce heuristic solver performance. Precomputation time for Zamboni patterns is categorically negligible and is minimal (several seconds) for both OTSP/SGTSP and mDCPP for suburban and less dense environments. However, due to its $O(|E|^2)$ greedy assignment of edges, very dense environments containing hundreds of segments result in precomputation times of several minutes on average for mDCPP. Though this too is likely insignificant in practice but may prove unsuitable for situations requiring frequent replanning. SGTSP precomputation time is generally better than OTSP given that the state space is entire edges rather than tessellated orbits, while increasing for very dense environments in which orbit elimination maintains low OTSP runtime. This is of no consequence, however, as in this density range Zamboni patterns are preferable.



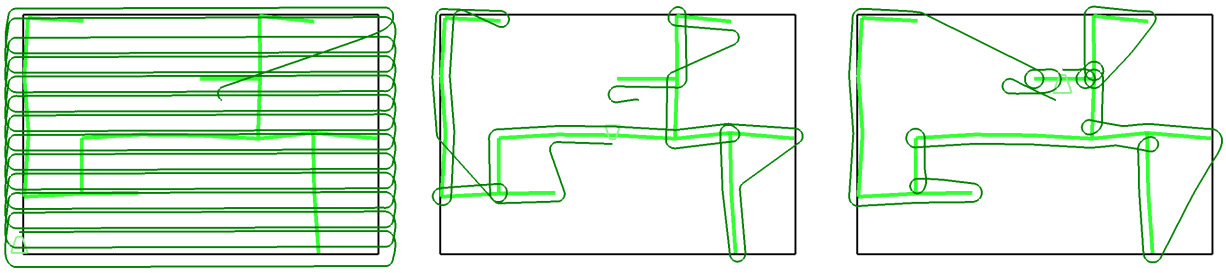
(a) Zamboni: 12,232s (b) mDCPP: 20,465s (c) OTSP: 16,177s

Figure 7: Sample execution results for three coverage algorithms in an urban-density environment. Zamboni patterns operate independently of the presence of any roads and cover all areas, even those not of interest. The mDCPP and OTSP strategies consider only roads, with the OTSP algorithm exhibiting better macroscopic path optimality owing to its use of a global solver but performs more maneuvering. Typically for this density, Zamboni outperforms the others.



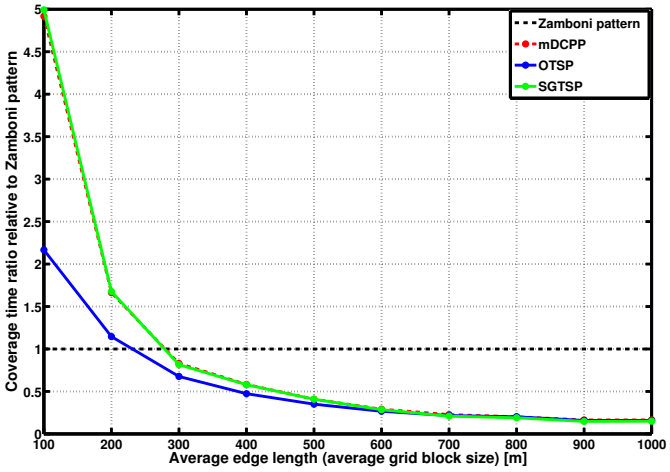
(a) Zamboni: 12,318s (b) mDCPP: 5,299s (c) OTSP: 4,490s

Figure 8: Sample execution results for three coverage algorithms in a suburban-density environment. Zamboni performs particularly poorly relative to the others because it is unvarying (up to random variation, exhibited here) in its path. Once again, OTSP shows better macroscopic optimality which is sufficient to outperform mDCPP, though some excessive maneuvering remains.

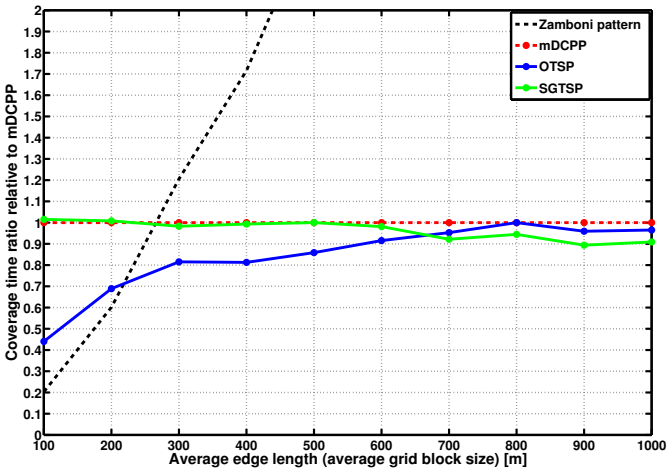


(a) Zamboni: 12,336s (b) mDCPP: 2,393s (c) OTSP: 2,426s

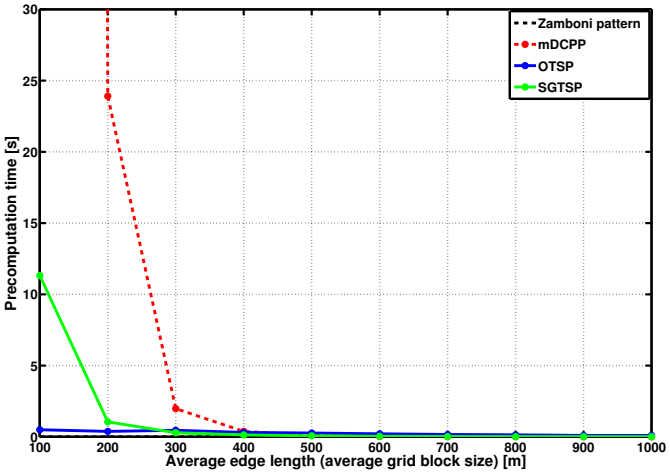
Figure 9: Sample execution results for three coverage algorithms in a rural-density environment. Here again, Zamboni performs particularly poorly relative to the others given the very small area of actual interest. Because it considers only full edges regardless of their length, mDCPP outperforms OTSP, which sequences more minute motions, though only by less than 2%.



(a) Coverage execution time relative to Zamboni



(b) Coverage execution time relative to mDCPP



(c) Precomputation time (Zamboni is negligible)

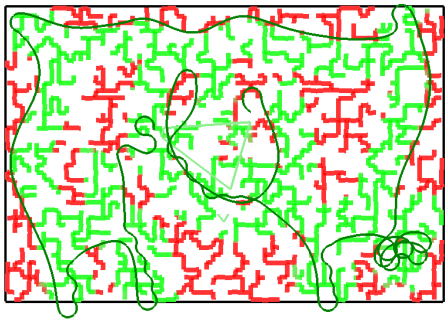
Figure 10: Coverage execution and precomputation time comparison between a Zamboni pattern and the greedy mDCPP competitor, orbit TSP (OTSP), and sweeping generalized TSP (SGTSP) algorithms for a single vehicle. Note that for very dense environments, simple a Zamboni patterns is most appropriate. For less dense ones, OTSP is best. Finally, for very rural environments, SGTSP is preferable.

These behaviors may be contrasted with traditional strategies for environment-aware and so-called information-theoretic trajectory planning. Executions of several simple instances of such are compared in Figures 11 and 12 for urban and rural environments. The first instance is a continuous control law that chooses an instantaneous steering angle corresponding to the probability gradient of a cellular Bayesian probability map maintained during the coverage search (and identical to that used for visualization), derived by convolving a Gaussian derivative kernel with the map at the center of the field of view with $\sigma = kR_{FOV}$ for some small integral k. The second is a single-step (greedy) horizon planner that searches among a set of discrete steering commands, which when held for a specified duration (zero-order hold control), maximizes the probability of detection in a similar Bayesian probability map within that next time-step, sampled at several points along the predicted trajectory. The third is a multi-step horizon planner performing an identical search, but in an action tree of steering angles. The hold duration for the latter two methods is chosen to be proportional to average road segment length so that the horizon is relevant for the scale of the environment (e.g. so that very short trajectory lookahead is not used in large sparse environments). Though Bourgault et al.[7] have demonstrated the efficacy of greedy planning for UAVs in Bayesian probability maps, its strength is primarily in producing semi-coordinated behaviors that quickly cover high-likelihood areas with minimal computation, while the primary weaknesses of such methods are that they will rarely provide *complete* coverage instead leaving scattered pockets of moderate target likelihood and are prone to getting stuck in local minima from which they will never reach the remaining pockets. Meanwhile, while multi-step horizon planning reduces such issues through non-minimum-phase behaviors enabled by lookahead, these too suffer inevitable myopia in sufficiently large environments and quickly become intractable with increasing horizon length. More advanced methods such as randomized planning are certainly available, however all fundamentally attempt to solve a continuous-space trajectory planning problem in possibly large environments and cannot offer any but asymptotic guarantees of complete coverage. In the sample executions shown, only multistep planning completes coverage successfully, and faster than an exhaustive Zamboni pattern.

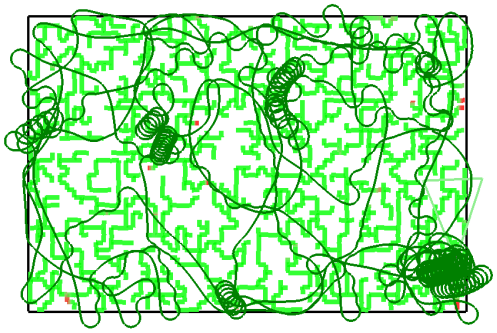
Coverage using multiple-vehicle teams was also evaluated. The primary property of note in expansion to multiple vehicles is the means by which coverage of the environment is distributed among them. For Zamboni patterns, simple axis-aligned geometric slices of the environment along its major axis are chosen, and each vehicle performs a Zamboni pattern in parallel within its slice. The mDCPP algorithm is intrinsically multi-vehicle via its deconfliction auction mechanism. For strategies involving a TSP solver, ad-hoc cooperation generated by the same geometric separation as used for Zamboni patterns in which each vehicle solves the coverage problem independently within its slice is one mechanism considered. The other is the use of a VRP solver, which simultaneously performs assignment and sequencing. However, strategy extensions requiring the use of a generalized TSP cannot make use of this given their formulation as a transformation to a standard TSP and are thus limited to ad-hoc cooperation. Example executions for three vehicles on a suburban-density environment are shown in Figure 13.

Experimental results for three vehicles are summarized in Figure 14. These indicate the continued preference for Zamboni patterns for sufficiently dense environments. Another trend similar trend to single-vehicle coverage is present in that OTSP again outperforms all others for sparse urban through suburban densities, while SGTSP is best for less dense environments. Of note here is that VRP-generated cooperation is moderately superior to ad-hoc geometric splitting at a fair substantial (but not intractable) increase in precomputation time. The weakness of ad-hoc cooperation relative to reasoned assignment may be seen explicitly in the better performance of the mDCPP strategy relative to SGTSP in dense environments: other than vehicle assignment, SGTSP should perform better.

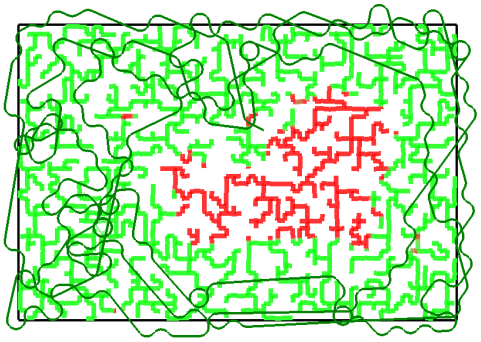
Finally, explicit comparison of the first two proposed strategy extensions – angle-discretized OTSP and pairwise OTSP – is presented in Figure 15 for three-vehicle coverage. The angle-discretized strategy always performs somewhat worse than the others, owing to several factors including misalignment between sampled poses and the vector of the road segment being swept, a moderately large state space reducing the quality of heuristic solutions, and sparse connectivity between states (given the clustering formulation) that further ill-conditions the graph provided to the heuristic solver. Capturing pairwise motion costs slightly improves performance, but at great computational cost due to the greatly increased problem size. Indeed, as environment density increases, performs actually worsens due to the inability of the heuristic solver to navigate the large graph, followed by total intractability at even higher densities. Overall, these results vindicate the use of the basic OTSP method over reasoned but complexity-introducing extensions, excepting SGTSP which these inspired.



(a) Probability gradient: initially captures a large amount of target probability rapidly (6,064s)



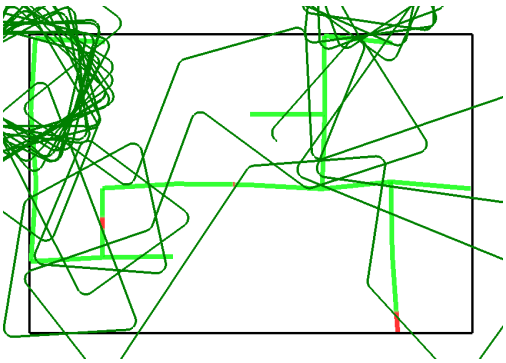
(b) Probability gradient: stuck in active local minimum, leaving small regions uncovered (15,000s)



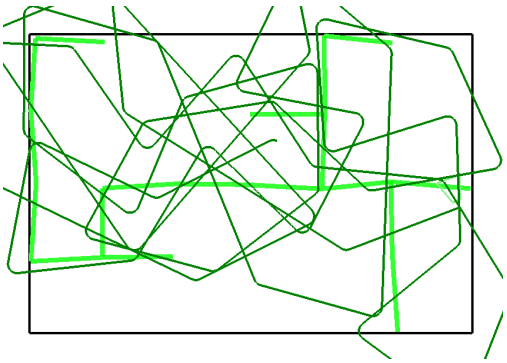
(c) Single-step (greedy) planning: stuck in local minimum (7,037s)

(d) Multi-step (horizon) planning: coverage complete (8,260s)

Figure 11: Sample execution results for three traditional environment-aware algorithms attempting to maximize short-term probability of detection in an urban-density environment. Of these, only multi-step planning (requiring 3.8s per planning step, which is infeasible for realtime execution) completes coverage.

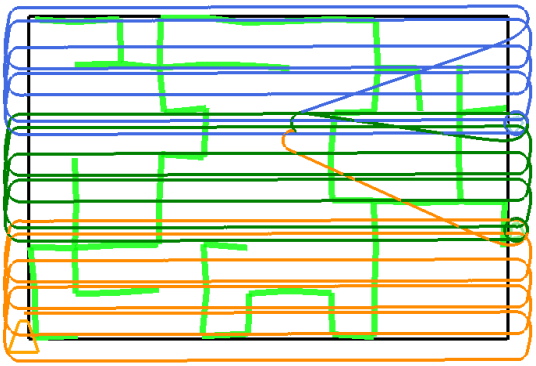


(a) Single-step (greedy) planning: stuck in local minimum (8,297s)

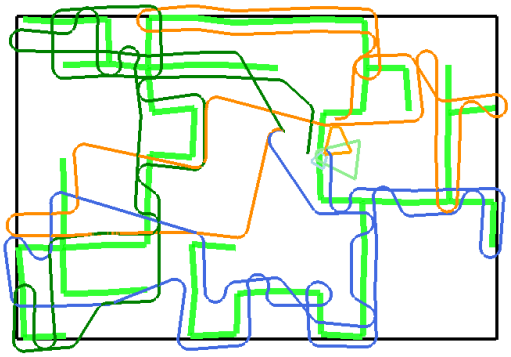


(b) Multi-step (horizon) planning: coverage complete (7,196s)

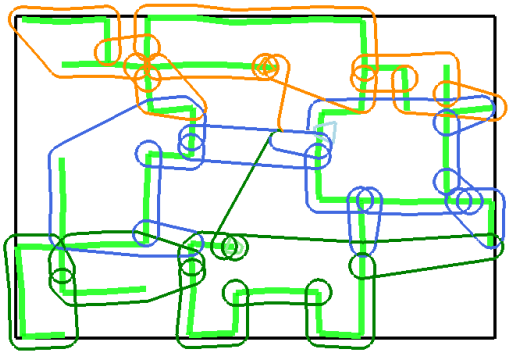
Figure 12: Sample execution results for two traditional environment-aware algorithms attempting to maximize short-term probability of detection in a rural-density environment. Of these, only multi-step planning completes coverage. The probability gradient rule is not even shown as it was unable to make any progress whatsoever, lacking a meaningful gradient at its start location.



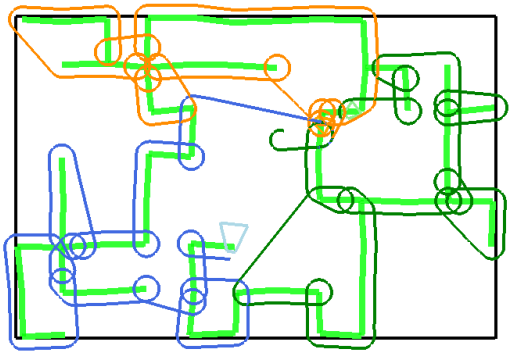
(a) Parallel Zamboni patterns: 4,561s



(b) mDCPP: 1,867s

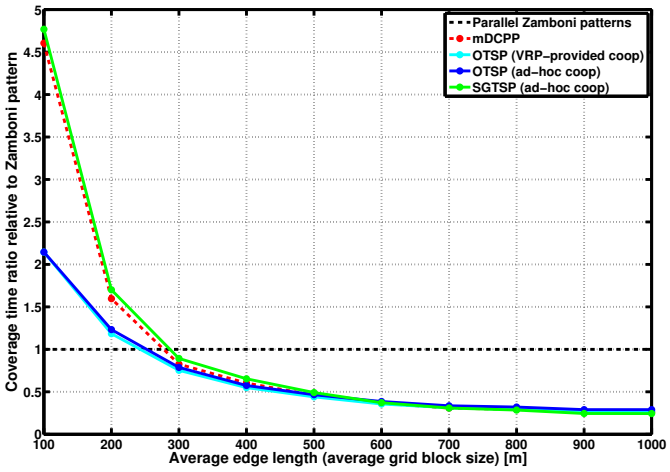


(c) OTSP with ad-hoc cooperation: 2,158s

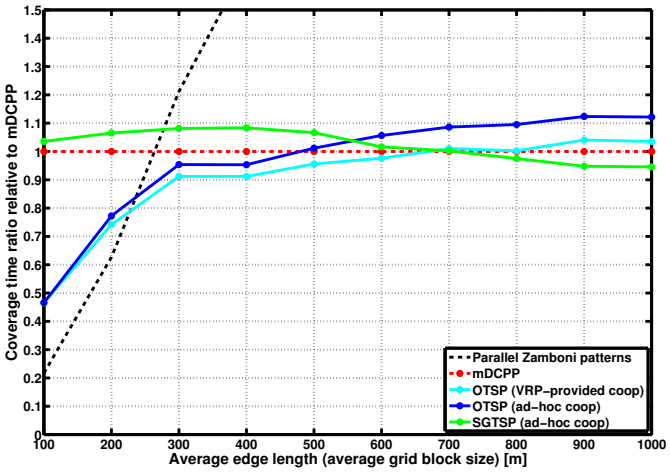


(d) OTSP with VRP-provided assignment: 1,825s

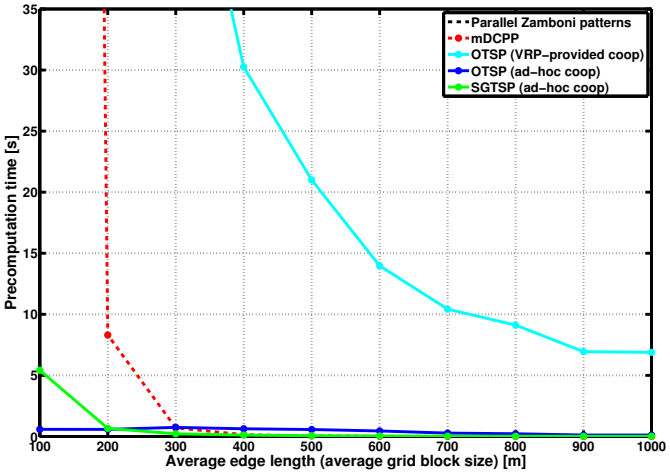
Figure 13: Sample execution results for three vehicles in a suburban-density environment comparing four coverage algorithms: parallel Zamboni patterns splitting the environment into equally-sized axis-aligned areas, the mDCPP algorithm which internally uses an auction-based mechanism for assignment, the basic OTSP strategy using ad-hoc cooperation in which three OTSP problems are generated from a geometric split of the environment identical to that used for the Zamboni patterns, and the basic OTSP strategy using a VRP solver internally that simultaneously solves the vehicle assignment and TSP sequence problems. This example is typical in that Zamboni patterns are unnecessarily exhaustive, mDCPP performs somewhat better than OTSP using ad-hoc cooperation, and OTSP using the VRP solver performs best. Note the clear benefit of reasoned area assignment over a hard geometric split and the continued superiority of OTSP over mDCPP for environments of such density with the effect of assignment removed by the VRP solver.



(a) Coverage execution time relative to Zamboni

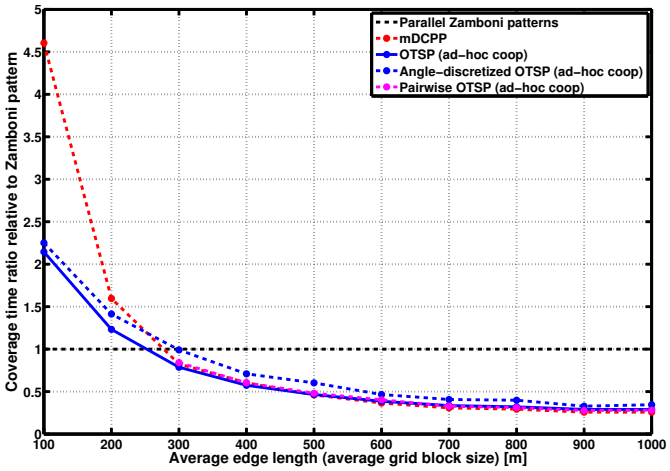


(b) Coverage execution time relative to mDCPP

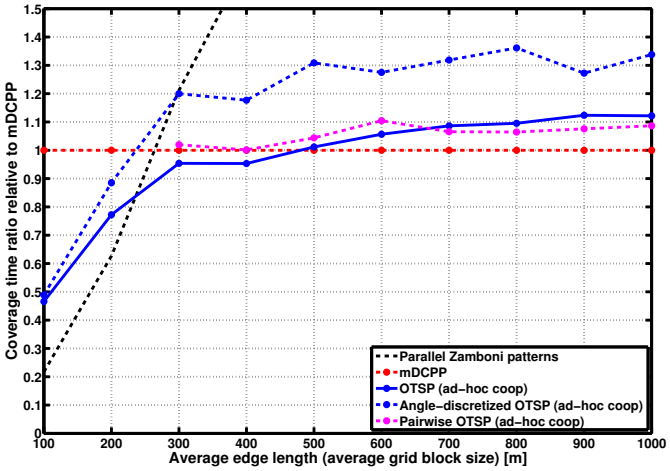


(c) Precomputation time (Zamboni is negligible)

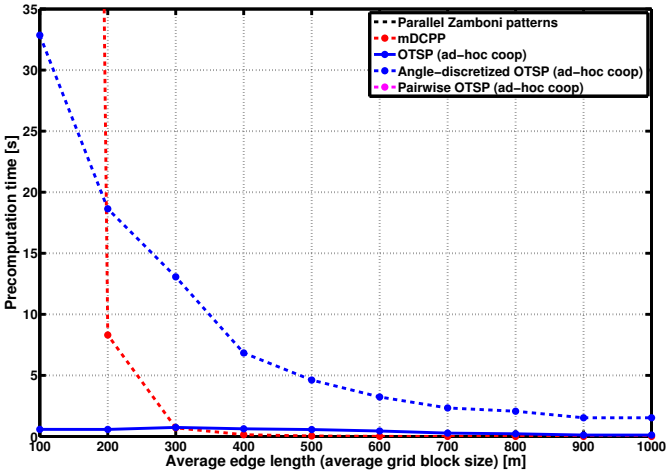
Figure 14: Coverage execution and precomputation time comparison between Zamboni patterns and the greedy mDCPP competitor, orbit OTSP (OTSP) using both a VRP solver and an ad-hoc spatial split of the environment into independent TSPs, and sweeping generalized TSP (SGTSP) algorithms for 3 UAVs running in parallel.



(a) Coverage execution time relative to Zamboni



(b) Coverage execution time relative to mDCPP (note that pairwise is intractable in dense environments)



(c) Precomputation time (Zamboni is negligible, while pairwise is non-realtime and lies beyond this graph)

Figure 15: Coverage execution and precomputation time comparison between the basic orbit TSP (OTSP) method and the two proposed extensions (angle-discretized and pairwise-orbit), alongside Zamboni patterns and the greedy mDCPP competitor algorithm for reference, for 3 UAVs running in parallel. Note that the angle-discretized strategy is always worse than the basic one and that the pairwise method is not always better than the basic one due to suboptimal heuristic solutions for the large resulting graph.

VII. Conclusions and Future Work

Efficient area coverage can be a particularly challenging task because it is one in which there is no goal location that must simply be reached by a vehicle, presenting a trajectory generation rather than path planning task. In sparse environments such as road networks, exhaustive and continuous-space planning methods for coverage may be grossly inefficient in either computation time or mission duration. Trajectory planning in a more easily represented space encoding only areas of interest can improve upon this. A new method for area coverage appropriate for sparse environments and motion-constrained vehicles such as fixed-wing UAVs was proposed and compared against exhaustive coverage and an existing road-sweeping strategy. For a wide range of environment densities that may be characterized as "suburban," this method outperforms both for single-vehicle coverage, whereas for dense environments ("urban") exhaustive search is more appropriate, and for very sparse ones ("rural") road-sweeping is best. For multiple vehicles, a similar pattern emerges. Proposed extensions were evaluated and shown to primarily increase complexity rather than performance, though these inspired a generalization of an existing greedy competitor that proved to be best overall for sparse environments. Given a novel environment, the density heuristic implied by these results may be applied to choose the most appropriate algorithm, or, since none has excessive precomputation time, one can simulate each to find the one with shortest expected mission duration.

Inspirations for future work primarily lie in the area of reasoned extension to multiple-vehicle teams. Current strategies include ad-hoc geometric splitting, an auction-based scheme, and the use of the VRP generalization of the TSP where applicable. Better strategies, particularly applied to the sweep-sequencing SGTSP method proposed, should be sought. One possible idea is a hybrid mDCPP-SGTSP strategy in which alternating steps of auctioning edges greedily and solving the TSP on each vehicle's current set of edges are taken. Another avenue for future exploration is better formalization of environmental properties such as density as a means for coverage algorithm selection, beyond merely simulating several in parallel prior to physical execution to make this choice.

VIII. Acknowledgements

The authors wish to thank Dr. Ben Grocholsky for thoughtful conversation and helpful suggestions.

References

[1]Ablavsky, V. and Snorrason, M., "Optimal Search for a Moving Target: A Geometric Approach," *AIAA Conference on Guidance, Navigation, and Control*, 2000.

[2]Quigley, M., Barber, B., Griffiths, S., and Goodrich, M. A., "Towards Real-World Searching with Fixed-Wing Mini-UAVs," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005.

[3]Enns, D., Bugajski, D., and Pratt, S., "Guidance and Control for Cooperative Search," *American Control Conference*, 2002.

[4]Beard, R. W. and McLain, T. W., "Multiple UAV Cooperative Search under Collision Avoidance and Limited Range Communication Constraints," *IEEE Conference on Decision and Control*, 2003.

[5]Maza, I. and Ollero, A., "Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms," *International Symposium on Distributed Autonomous Robotic Systems*, June 2004.

[6]Girard, A. R., Howell, A. S., and Hedrick, J. K., "Border Patrol and Surveillance Missions using Multiple Unmanned Air Vehicles," *IEEE Conference on Decision and Control*, 2004.

[7]Bourgault, F., Furukawa, T., and Durrant-Whyte, H., "Coordinated decentralized search for a lost target in a Bayesian world," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[8]Polycarpou, M. M., Yang, Y., and Passino, K. M., "A Cooperative Search Framework for Distributed Agents," *IEEE International Symposium on Intelligent Control (ISIC)*, 2001.

[9]Tisdale, J., Kim, Z., and Hedrick, J. K., "An Autonomous System for Cooperative Search and Localization using Unmanned Vehicles," *AIAA Conference on Guidance, Navigation, and Control*, 2008.

[10]Geyer, C., "Active Target Search from UAVs in Urban Environments," *IEEE International Conference on Robotics and Automation (ICRA)*, 2008.

[11]Mathew, G. and Mezic, I., "Spectral Multiscale Coverage: A Uniform Coverage Algorithm for Mobile Sensor Networks," *IEEE Conference on Decision and Control*, 2009.

[12]Edmonds, J. and Johnson, E. L., "Matching, Euler Tours, and the Chinese Postman," *Mathematical Programming*, Vol. 5, No. 1, 1973, pp. 88–124.

[13]Thimbleby, H., "The directed Chinese Postman Problem," *Software: Practice and Experience*, Vol. 33, No. 11, 2003, pp. 1081–1096.

[14]Oh, H., Shin, H., Tsourdos, A., White, B., and Silson, P., "Coordinated Road Network Search for Multiple UAVs Using Dubins Path," *Advances in Aerospace Guidance, Navigation and Control*, 2011, pp. 55–65.

[15]Oh, H., Kim, S., Tsourdos, A., and White, B., "Cooperative Road-Network Search Planning of Multiple UAVs Using Dubins Paths," *AIAA Conference on Guidance, Navigation, and Control*, 2011.

[16]Applegate, D., Bixby, R., Chvatal, V., and Cook, W., "Concorde TSP Solver," `http://www.tsp.gatech.edu/concorde.html`, December 2011.

[17]Savla, 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.

[18]Le Ny, J., *Performance Optimization for Unmanned Vehicle Systems*, Ph.D. thesis, Massachusetts Institute of Technology, 2008.

[19]Faied, M., Mostafa, A., and Girard, A., "Vehicle Routing Problem Instances: Application to Multi-UAV Mission Planning," *AIAA Conference on Guidance, Navigation, and Control*, 2010.

[20]Dille, M., Grocholsky, B., and Singh, S., "Persistent Visual Tracking and Accurate Geo-Location of Moving Ground Targets by Small Air Vehicles," *AIAA Infotech@Aerospace Conference*, March 2011.

[21]Dille, M., Grocholsky, B., Nuske, S., Moseley, M., and Singh, S., "Air-Ground Collaborative Surveillance with Human Portable Hardware," *AUVSI Unmanned Systems North America*, August 2011.

[22]Dubins, L., "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.

[23]Tisdale, J., Durrant-Whyte, H., and Hedrick, J. K., "Path Planning for Cooperative Sensing Using Unmanned Vehicles," *ASME International Mechanical Engineering Congress and Exposition*, November 2007.

[24]Frew, E. W. and Lawrence, D., "Cooperative stand-off tracking of moving targets by a team of autonomous aircraft," *AIAA Guidance, Navigation, and Control Conference*, 2005.

[25]McGee, T. G., Spry, S., and Hedrick, J. K., "Optimal path planning in a constant wind with a bounded turning rate," *AIAA Conference on Guidance, Navigation, and Control*, 2005.

[26]Gibson, M., *Clusters and covers: geometric set cover algorithms*, Ph.D. thesis, University of Iowa, 2010.

[27]Isaacs, J. T., Klein, D. J., and Hespanha, J. P., "Algorithms for the Traveling Salesman Problem with Neighborhoods Involving a Dubins Vehicle," *American Control Conference*, 2011.

[28]Noon, C. E. and Bean, J. C., "An Efficient Transformation of the Generalized Traveling Salesman Problem," Tech. Rep. 91-26, University of Michigan, Ann Arbor, MI, October 1991.

[29]Kumar, R. and Li, H., "On Asymmetric TSP: Transformation to Symmetric TSP and Performance Bound," Tech. rep., Iowa State University, Ames, IA, 1995.

[30]Jonker, R. and Volgenant, T., "Transforming Asymmetric into Symmetric Traveling Salesman Problems," *Operations Research Letters*, Vol. 2, No. 4, November 1983, pp. 161–163.

[31]AeroVironment Inc., "Raven Product Data Sheet," `http://www.avinc.com/downloads/Raven_Domestic_1210.pdf`, December 2010.