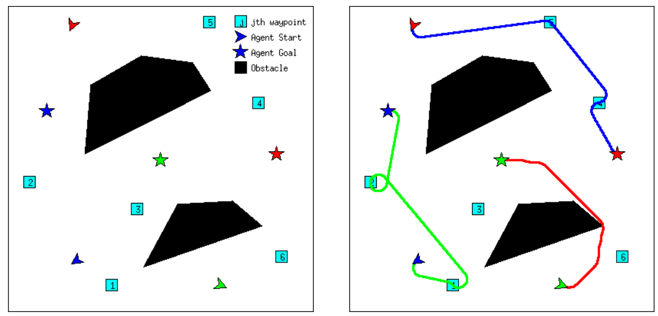
2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Planning for Opportunistic Surveillance with Multiple Robots

Dinesh Thakur*, Maxim Likhachev†, James Keller*, Vijay Kumar*, Vladimir Dobrokhodov‡, Kevin Jones‡, Jeff Wurz‡ and Isaac Kaminer‡
* GRASP Laboratory, University of Pennsylvania, Philadelphia PA
† Robotics Institute, Carnegie Mellon University, Pittsburgh, PA
‡ Department of Mechanical and Aerospace Engineering, Naval Postgraduate School, Monterey, CA

***Abstract*— We are interested in the multiple robot surveillance problem where robots must allocate waypoints to be visited among themselves and plan paths through different waypoints while avoiding obstacles. Furthermore, the robots are allocated specific times to reach their respective goal locations and as a result they have to decide which robots have to visit which waypoints. Such a problem has the challenge of computing the allocation of waypoints across robots, ordering for these waypoints and dynamical feasibility of the paths between waypoints. We present an algorithm that runs a series of graph searches to solve the problem and provide theoretical analysis that our approach yields an optimal solution. We present simulated results as well as experiments on two UAVs that validate the capability of our algorithm. For a single robot, we can solve instances having 10-15 waypoints and for multiple robots, instances having five robots and 10 waypoints can be solved.**



(a) Environment with three robots and six waypoints to be visited.

(b) Planned paths satisfying start and goal constraints ensuring maximum waypoints are visited.

Fig. 1: Environment with three robots and six waypoints.

I. Introduction

Consider multiple robots available at different locations that are needed to visit a number of waypoints for reconnaissance. Each waypoint can be visited at most once and because each robot has limited fuel/flight-time capabilities, it may not be possible to visit all waypoints. This is an example of Opportunistic Surveillance and also known as the Team Orienteering Problem (TOP) in which multiple robots are required to maximize the number of waypoints visited, subject to an upper bound on the total length (or time) of travel for each robot. The Orienteering Problem (OP) is the single robot version of the TOP.

An example of the problem with three robots with their corresponding goals and six waypoints that the robots must visit is shown in Figure 1a. The black areas indicate obstacles or no-fly zones. The planner has to decide which waypoints can be visited by which robots while proceeding to and reaching their respective goal locations within their given time limits. Thus, the problem has three computational challenges: computing the allocation of waypoints across all robots, computing the order of visiting waypoints for each robot and computing paths between any two pairs of waypoints. Typical existing approaches assume the costs of transitions between waypoints are known and do not address the dynamic constraints of the robots [1]. However, the time to compute the actual transitions can be significant if there are dynamic constraints such as constraints on the minimum turning radii of the robots. In this paper, we propose an approach that folds the computations of transitions into the process of solving the TOP itself. To address the computational challenges, we formulate the problem as a three-tier graph search algorithm. In particular, the algorithm solves waypoints allocation across robots, ordering of the waypoints for each robot and path planning problem all simultaneously while providing rigorous guarantees on completeness and optimality. By interleaving the search across all the three levels we are able to make it scalable for up to five robots and ten waypoints. Figure 1b shows the output of our algorithm for the example given in Figure 1a. Two robots cover two waypoints each within its alloted time while one robot covers none. All of these generated paths respect the minimum turning radius constraints of the robots. In particular these paths are curves in three dimensional space parameterized by coordinates $\{x, y, \theta\}$. The paper describes the algorithm, analyzes its properties and presents experimental evaluation in simulation and on a team of two physical fixed-wing aerial vehicles.

II. Related Work

The Orienteering Problem (OP) and the Team Orienteering Problem (TOP) are extensively studied in the Operations Research community. The OP is also known as the Selective Traveling Salesman Problem (STSP), Maximum Collection Problem, Bank Robber Problem, Prize Collecting TSP, among other problems. A detailed survey of the existing OP and TOP literature is given in [1].

The OP is NP-hard [2] and only a few exact algorithms have been proposed. With the Branch-and-Bound approach, problems up to 150 locations have been solved [3], [4].

Instances up to 500 locations can be solved optimally with the Branch-and-Cut approach proposed in [5], [6].

Most of the OP research has mainly focused on heuristic approaches. A stochastic (S-Algorithm) and a deterministic algorithm (D-Algorithm) is proposed in [7]. A center-of-gravity heuristic is developed in [2], [8] and five step heuristic is introduced in [9]. More heuristic algorithms have been developed to solve the OP [10], [11], [12], [13], [14], [15].

An exact algorithm to solve the TOP based on column generation is presented in [16]. A branch-and-price approach to solve problems with 2-4 team members and up to 100 locations is proposed in [17].

The first published TOP heuristic was developed in [18] and resembles the five-step heuristic for the OP. A tabu search heuristic procedure is developed in [19]. Two variants of a tabu search heuristic and a slow and fast Variable Neighbourhood Search (VNS) algorithm is given in [20]. Four variants of an ant colony optimization approach for the TOP is developed in [21]. More recent heuristics approaches are given in [22], [23], [24].

In all of the above work the costs of transitions between waypoints are assumed to be given. However, computing paths that take dynamic constraints of the robots can be expensive. In fact, we cannot assume that we can plan optimal paths between all pairs of waypoints for every robot. In our work, we do not assume the costs of transitions are given and compute them on the fly instead. Our algorithm optimizes the *overall* planning time by interleaving the search for optimal waypoint assignment, ordering of the waypoints and planning dynamically feasible paths between waypoints.

Our approach is also related to [25] in the sense that this work also uses multi-level graph search for a multi-agent team of robots. Their approach guarantees optimality only under certain conditions though while we guarantee optimality under all conditions.

III. Algorithm

A. Problem Formulation

We assume we are given N robots with corresponding start and goal coordinates S_i and G_i respectively, $1 \leq i \leq N$.

Additionally, we are given M waypoints W_j, $1 \leq j \leq M$ of interest. The robot start and goal locations and the waypoint locations are given by coordinates $\{x, y, \theta\}$. Each of the N robots has to travel from S_i to G_i within the alloted time T_{MAX_i}. The goal of the planner is to compute N dynamically feasible paths π_i that cover as many waypoints as possible and minimize the cumulative cost $\sum_{i=1}^{N} c(\pi_i)$ with a constraint that $c(\pi_i) \leq T_{MAX_i} \forall \pi_i$. We assume the cost $c(\pi_i)$ is the time it takes to traverse π_i.

B. Overview of the approach

Our approach is based on a graph-based representation. We construct and search for a solution three levels of graphs: a top level graph G_T, mid level graph G_M and a low level graph G_L. Figure 2 shows the three graph levels. The graph G_T encodes which waypoints have to be assigned to which robot. Each edge in this graph G_T corresponds to assigning an additional waypoint to a robot. Given a single robot R_i, the mid level graph G_{M_i} represents the maximum number of waypoints that can be covered within the time limit T_{MAX_i}. Each edge in the graph G_{M_i} corresponds to moving from one waypoint to another. The cost of the edge is found by searching the graph G_L. Finally, the graph G_L represents the problem of finding a least-cost path that corresponds to going from one waypoint to another while avoiding obstacles, given the dynamic constraints of the robot. A detailed description of the construction of the three graphs is given in Section III-C.

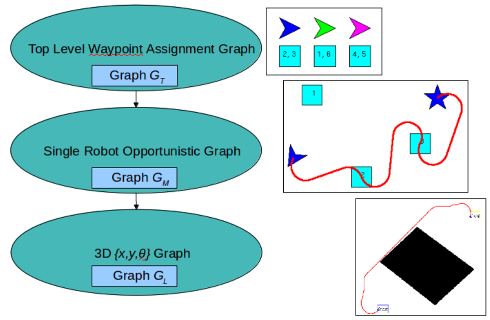


Fig. 2: Overview of the approach showing the three levels of representation.

The three graphs are constructed on the fly as the search progresses. The search interleaves searching the top level, the mid level and the low level graph. Such interleaving helps to focus the search efforts of the low level search on the problems that are relevant to the mid level search and focus the mid level search to the problems that are relevant to the top level search. Another advantage of breaking down the search into three graphs is that the solutions can be reused and the overall memory footprint of the search is smaller. The graph search itself is described in Section III-D.

C. Three level graph representation of the problem

1) Graph G_T: Each node $q \in V(G_T)$ is defined as $\{\beta\}$, a vector of M elements where M is the number of waypoints. Each element represents a robot assigned to the corresponding waypoint. For example, if $M = 3$ and $N = 2$, the node $q = \{(2,1,2)\}$ represents that the waypoints W_1 and W_3 are assigned to robot R_2 while W_2 is assigned to robot R_1. 0 represents an unassigned waypoint. The start node is always $q_{start} = \{(0,0,\cdots)\}$, none of the waypoints are assigned.

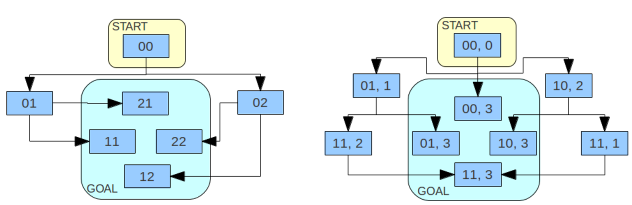
Each edge in G_T corresponds to assigning a single waypoint, the first waypoint that is still unassigned, to one of the robots. We use $succ(q)$ to denote the set of successor states of the state $q \in V(G_T)$. For example, consider $M = 3$, $N = 2$, $q = \{(0,0,1)\}$, and $q, q_1, q_2, q_3, q_4 \in V(G_T)$. The valid successors of q are $q_1 = \{(0,1,1)\}$ and $q_2 = \{(0,2,1)\}$, while $q_3 = \{(1,0,1)\}$ and $q_4 = \{(2,0,1)\}$ are not successors of q, *i.e.* $q_1, q_2 \in succ(q)$, while $q_3, q_4 \notin succ(q)$. Figure 3a shows an example of a top level graph transitions with $N = 2$ and $M = 2$.

The cost of the transition from one state to another in G_T corresponds to the cost of the least-cost path obtained from the mid level graph G_M. Since q and q' only differ by a single element, the corresponding cost is the cost of assigning an additional waypoint to the robot. If $q = \{\beta\}$, $q' = \{\beta'\}$, $q' \in succ(q)$ and the ith robot is assigned from q to q', then $c(q,q') = c^*_{q'}(n_{S_i}, n_{G_i}) - c^*_q(n_{S_i}, n_{G_i})$. Here n_{S_i} and n_{G_i} represent the mid level start and goal state for the ith robot and $c^*_{q'}(n_{S_i}, n_{G_i})$ represents the cost of optimal path in G_M with waypoints available for the ith robot according to the node q'. Similarly, $c^*_q(n_{S_i}, n_{G_i})$ represents the cost of optimal path in G_M with waypoints available for the ith robot according to the node q. Since q is a predecessor state, this cost is already known. As explained in the Section III-D.1, an optimal path through G_T corresponds to an optimal assignment of the waypoints to the robots. Let k be the total number of unvisited waypoints across all agents in q'. If $q' = q_{goal}$, we add a cost $k \times P_T$ in addition to the transition cost $c(q, q_{goal})$, where $P_T = \sum_{i=1}^{N} T_{MAX_i} + 1$ is the penalty for not visiting each waypoint. The section III-D.1 explains the choice of such penalty.

$$c(q,q') = \begin{cases} c^*_{q'}(n_{S_i}, n_{G_i}) - c^*_q(n_{S_i}, n_{G_i}), \; q' \neq q_{goal} \\ c^*_{q'}(n_{S_i}, n_{G_i}) - c^*_q(n_{S_i}, n_{G_i}) + k \times P_T, \text{ otherwise} \end{cases} \tag{1}$$

2) Graph G_M: The graph G_M encodes the path for a single robot that goes through as many waypoints as possible that were assigned to it while minimizing time. Each node $n \in V(G_M)$ is defined as $\{\alpha, \Omega\}$, α is a vector of M bits where a 0 bit indicates the corresponding waypoint is unvisited and 1 bit indicates it is visited. Ω is the index of the waypoint where the robot is currently at.

In our approach at the mid level, the planner needs to know the waypoints that have been visited, the current robot location and the time taken to reach the location. We use a model very similar to one that is presented in [25]. To model the waypoints visited and to have a search graph we use a state coordinate called α which is a variable that represents a binary number consisting of M bits (for M waypoints). Each bit indicates whether the corresponding waypoint has been visited or not. Similar to [25], for notational convenience we define a function B such that $\alpha = B_M(P_1, P_2, \cdots, P_k)$ is a M-bit long binary number with 1's at positions $P_1, P_2, \cdots, P_k$, and 0's at the rest of the positions. Thus, $B_5(2,4,5) = 11010$ and $B_3(1) = 001$. $B_M(P_1, P_2, \cdots, P_k)$ represents the state in which only the waypoints $P_1, P_2, \cdots, P_k$ have been visited. Since at the start location no waypoints are visited, $\alpha_{start} = B_M()$. For the goal state, α_{goal} is not unique because the number of waypoints visited can be anything from 0 to M. If all the waypoints have been visited $\alpha_{goal} = B_M(1,2,...,M)$, if none are visited $\alpha_{goal} = B_M()$.



(a) Top level graph G_T with two robots, two waypoints and state transitions. **(b)** Mid level graph G_M with two waypoints and state transitions.

Fig. 3: Top level and mid level graphs.

Consider $B_5(2,4,5)$. This function does not provide information on the current location of the robot. It just tells us that waypoints 2, 4 and 5 have been covered. In order to model the current location of the robot and to have a unique goal state, an additional parameter is required. We use a state coordinate Ω to represent the current location of the robot. Ω can be a start location denoted by $\Omega = 0$, a goal location denoted by $\Omega = M+1$ or any waypoint location W_j given as $\Omega = j$, where $1 \leq j \leq M$. Thus each node $n \in V(G_{M_i})$ is represented as $\{\alpha, \; \Omega\}$. If $M = 5$, $n = \{B_5(2,4,5), 4\}$ represents that the robot has covered waypoints 2, 4, 5 and its current location is 4. Note, $n_1 = \{B_5(2,4,5), 4\}$ is similar to $n_2 = \{B_5(4,2,5), 4\}$ as the order of visiting the waypoints does not matter.

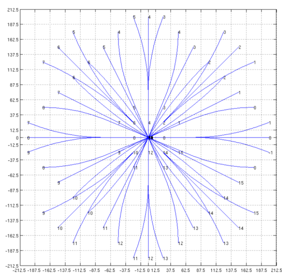
Since only one waypoint is visited at a time, any two vertices in G_M are connected *iff* they only differ by a single bit in α. The edge direction is from a vertex with a lower value of α to that of a higher value and different α. We use $succ(n)$ to denote the set of successor states of state $n \in V(G_M)$. If, for example, $M = 3$, and $n = \{B_3(1), 1\}$, the valid transitions from n are $n_1 = \{B_3(2,1), 2\}$ and $n_2 = \{B_3(3,1), 3\}$, while $n_3 = \{B_3(2), 2\}$ and $n_4 = \{B_3(2,1), 1\}$ are invalid transitions *i.e.* $n_1, n_2 \in succ(n)$ while $n_3, n_4 \notin succ(n)$, where $n, n_1, n_2, n_3, n_4 \in V(G_M)$. Figure 3b shows an example of a mid level graph transitions with $M = 2$.

In the mid level graph, if $n = \{\alpha, \Omega\}$, $n' = \{\alpha', \Omega'\}$ and $n' \in succ(n)$ then $c(n,n') = c^*(s_\Omega, s'_{\Omega'})$. s_Ω and $s'_{\Omega'}$ represent the low level state of the two waypoints corresponding to Ω and Ω'. Thus, the cost of the transition from one state to another in G_M is the cost of the least-cost path obtained from the low level graph G_L. Every state where $\Omega = M+1$ is considered to be a goal *i.e.* a robot is allowed at any point to go directly to its goal location. If $n' = n_{goal}$ we define $c(n, n_{goal}) = c^*(s_\Omega, s'_{\Omega'}) +$ (Number of waypoints not visited in n) $\times P_M$, where $P_M = T_{MAX} + 1$ is the penalty for not visiting each waypoint. This guarantees that the optimal path through G_M corresponds to visiting as many waypoints as possible within T_{MAX} and spending minimum amount of time to do it. Thus,

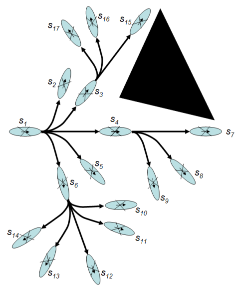
$$c(n,n') = \begin{cases} c^*(s_\Omega, s'_{\Omega'}), \text{ if } n' \neq n_{goal} \\ c^*(s_\Omega, s'_{\Omega'}) + (\#\text{ waypoints not visited in } n) \times P_M \end{cases} \tag{2}$$

In addition, any transition from n to n' is invalidated whenever time to reach n' exceeds T_{MAX_i}. The time to reach n' is given by its g value as explained later in Section III-D.2

3) Graph G_L: The formulation of graph-based planning involves discretization of the configuration space into a set of states, representing configurations, and transitions between these states, where every transition represents a feasible path. We discretize the graph with states $s \in V(G_L)$ as $\{x, y, \theta\}$, where x, y represent the position of the robot and θ represents the orientation of the robot. Once G_L is formed by the discretization of the configuration space into a set of states, connections between states represent short feasible paths. This essentially corresponds to lattice based



(a) Motion primitives at different orientations.



(b) Replication of motion primitives.

Fig. 4: The low-level graph, G_L.

graphs [26], [27] which are well suited to planning for non-holonomic robotic systems such as passenger vehicles and UAVs. Figure 4a shows the motion primitives we used in our experiments for all the 16 possible headings of the robot. Since we used UAVs for our experiments, the discretization and transitions between states were designed based on their kinematic and dynamic constraints. Figure 4b shows how the graph is constructed by replicating motion primitives during the search.

For $s' \in succ(s)$, the edge $s \rightarrow s' \in E(G_L)$ is associated with a strictly positive cost $c(s,s')$ which is the cost of the action that connects s to s'. The cost of each transition is given by the time it takes to execute the corresponding motion primitive.

D. Graph searches at three levels

The three graph searches are interleaved to generate a provably optimal solution w.r.t. discretization.

1) Top level Search: The Pseudo-code 1 explains the *Multi Robot Opportunistic Path Refinement* algorithm. The main loop is an $A^\star$ search on the graph G_T which finds a least-cost path from q_{start} to q_{goal}. $A^\star$ maintains g-values for each state it has visited so far. $g(q)$ is always the cost of the best path found so far from q_{start} to q. The code is initialized by inserting q_{start} in *OPEN* which is a priority queue and setting $g(q_{start}) = 0$. $A^\star$ prioritizes the states that are chosen from *OPEN* based on their f-values, $f(q) = g(q)$. The code removes states from *OPEN* and expands them using lines 7 through 13. This repeats until q_{goal} is expanded or there are no more states left to expand in *OPEN*.

During the evaluation of a state q in lines 8 through 10, costs of transitions are assigned to each of the successor state q'. The transition cost from q to q' is obtained from a call to the mid level graph search (line 8). The exact details of the mid level search is given in the next section.

To drive the search through other unassigned waypoints the cost of transition to a goal state must be greater than the largest path cost. Any path in the graph G_T has a cost that is bounded by $\sum_{i=1}^{N} T_{MAX_i}$ as each robot R_i has a limited travel time T_{MAX_i}. We assign the cost of reaching a goal state q_{goal} from any other state q as $c(q,q_{goal})$ = $c^*_{qoal}(n_{S_i},n_{G_i}) - c^*_{q}(n_{S_i},n_{G_i})$ + (number of waypoints not visited across all agents in q_{goal})$\times P_T$, where $P_T = \sum_{i=1}^{N} T_{MAX_i} + 1$. Thus, the penalty for a path that has more unvisited waypoints is greater than penalty for the path that has minimum unvisited waypoints. Since the penalty is greater than the sum of the time to travel for all agents, the path with more unvisited waypoints is suboptimal. The theoretical proof given in Subsection IV further shows why this choice of cost function ensures that the search minimizes the number of unvisited waypoints.

Pseudo-code 1 Multi Robot Opportunistic Path Refinement

1: **procedure MultiRobotInterleave()**
2: $OPEN = \{q_{start}\}$
3: $g(q_{start}) = 0$
4: $g(q) = \infty$, $\forall q \in V(G_T)$ and $q \neq q_{start}$
5: **while** q_{goal} not expanded **do**
6: remove q with smallest $f(q)$ from *OPEN*
7: **for** each successor q' of q **do**
8: $c(q,q')$ = **SingleRobotInterleave**(waypoints for robot i according to state q') - $c^*_q(n_{S_i},n_{G_i})$
9: **if** $q' == q_{goal}$ **then**
10: $c(q,q')$ += (# waypoints not visited across all agents in q') $\times$ P_T
11: **if** $g(q') > g(q) + c(q,q')$ **then**
12: $g(q') = g(q) + c(q,q')$
13: insert q' in *OPEN* with $f(q') = g(q')$;

2) Mid level Search: The Pseudo-code 2 explains the *Single Robot Opportunistic Path Refinement* algorithm. The main loop is $A^\star$ search on the graph G_M which finds a least-cost path from n_{start} to n_{goal}.

During the expansion of a state n in lines 8 through 10, costs of transitions are assigned to each of the successor state n'. We use the time to traverse Dubins paths as cost estimates for this transition denoted as $d^*(n,n')$. The Dubins path cost is the optimal path cost in an obstacle free environment. If the Dubins path traverses through an obstacle, the cost can be dramatic underestimate. We then call the low level graph search and get the exact path cost ($c^*(s_n, s'_{n'})$) and assign it to the transition cost from n to n'. Figure 5 shows a case when the low level graph search is called.

To drive the search through as many waypoints as possible the cost of transition to a goal state must be greater than the largest path cost. We assign the cost of reaching goal state n_{goal} from any other state n as $c(n,n_{goal})$ = $c^*(s_n, s_{n_{goal}})$ + (number of waypoints not visited in n)$\times P_M$, where $P_M = T_{MAX} + 1$. To understand why this ensures that the search tries to maximize the number of visited waypoints consider the fact that any path through the graph G_M will have the cost that is bounded by T_{MAX}. As a result any path that does not go through as many waypoints as possible within the time T_{MAX} will have a penalty that is higher than the path that goes through the maximum number of waypoints. Since the penalty is greater than the time to travel itself, it will be a suboptimal path while the $A^\star$ search always finds the optimal path. The theoretical proof given in Subsection IV further explains why this choice of cost function ensures the search maximizes the number of waypoints covered.

Since cost is defined by time and the search is optimal $g(n) + c(n,n')$ represents the time it takes to reach the state

Pseudo-code 2 Single Robot Opportunistic Path Refinement

1: **procedure SingleRobotInterleave(waypoints)**
2: $OPEN = \{n_{start}\}$
3: $g(n_{start}) = 0$
4: $g(n) = \infty, \forall n \in V(G_M)$ and $n \neq n_{start}$
5: **while** n_{goal} not expanded **do**
6:   remove n with smallest $f(n)$ from $OPEN$
7:   **for** each successor n' of n **do**
8:     $c(n,n') = d^*(n,n')$
9:     **if** dubin's path goes through obstacle **then**
10:       $c(n,n') = c^*(s_n, s_{n'})$ //from low-level search
11:     $C(n,n') = c(n,n')$
12:     **if** $n' == n_{goal}$ **then**
13:       $C(n,n')$ += (# waypoints not visited in n) $\times$ P_M
14:     **if** $g(n) + c(n,n') < T_{MAX}$ and $g(n') > g(n) + C(n,n')$ **then**
15:       $g(n') = g(n) + C(n,n')$
16:       insert n' in $OPEN$ with $f(n') = g(n')$;
17: **return** $g(n_{goal})$ - (# waypoints not visited) $\times$ P_M

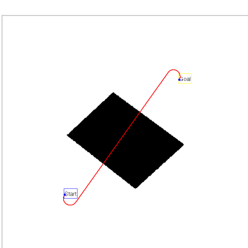
n' through the state n. Using this, lines 14 through 16 of the code ensure that the successor state n' being pushed in $OPEN$ does not violate the alloted maximum time condition.

3) Low level Search: The $A^\star$ search is perhaps one of the most popular methods for doing a graph search that finds a least-cost path from a given initial state to a goal state [28]. It utilizes a heuristic to focus the search towards the most promising areas of the search space. While highly efficient, $A^\star$ aims to find an optimal path which may not be feasible given time constraints and the dimensionality of the problem.

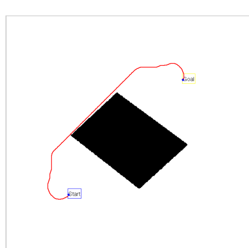
The heuristic of a state $h(s)$ is an estimate of the cost of a shortest path from current state s to the goal state s_{goal}. For the $A^\star$ to be optimal the heuristics must be admissible and consistent. For a heuristic to be admissible it must not overestimate the distance to the goal, $h(s) \leq c^*(s, s_{goal})$. A heuristic is consistent if $h(s) \leq c(s, succ(s)) + h(succ(s)), \forall s \neq s_{goal}$ and $h(s_{goal}, s_{goal}) = 0$.

An informed heuristic plays a major role in the $A^\star$s behavior. The lower $h(s)$ is, the more states the $A^\star$ expands, making it slower.

The most common heuristic used is Euclidean distance. Since the low level graph G_L is a 3D graph with states $\{x, y, \theta\}$, the euclidean distance is a significant underestimate for the $A^\star$ search. This leads to more expansions and more

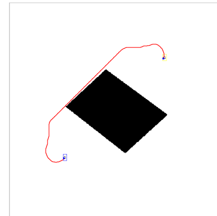


(a) Dubins path traversing through an obstacle.

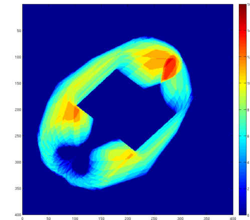


(b) Low level graph search using $A^\star$.

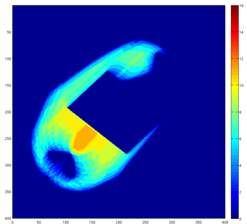
Fig. 5: Example showing mid level interleaving search



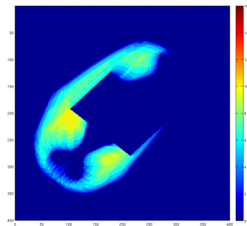
(a) A simple environment along with the optimal solution.



(b) heuristic = h_{2D_obs}



(c) heuristic = h_{dubins}



(d) heuristic=$max(h_{2D\_obs}, h_{dubins})$

Fig. 6: Visualization showing the total number of expanded states with different heuristics.

planning time. Instead we use Dubins optimal path as the heuristic for the graph search. The Dubins optimal path is an exact estimate of the path distance from s to s_{goal} in an obstacle free environment.

Given a non holonomic vehicle with a constraint on the minimum turning radius, constant forward speed and s_{start} and s_{goal}, [29] geometrically established that the optimal path (without obstacles) is one of the six possible configurations. All the paths are composed of three segments: CCC or CSC ($C \rightarrow$ curved at maximum curvature, $S \rightarrow$ straight). Each segment is a constant action over an interval of time. The shortest path between any two configurations can always be characterized by one of the six configurations $LSL, RSR, LSR, RSL, LRL, RLR$. In the S segment the vehicle drives straight ahead. During the L and R segments, the vehicle turns as sharply as possible to the left or to the right respectively.

Three different heuristics (h_{2D_obs}, h_{dubins} and $max(h_{2D\_obs}, h_{dubins})$) were used to run the numerical experiments. h_{2D_obs} is precomputed by running a 2D Dijkstra's search on x, y grid, h_{dubins} is the Dubins path heuristic and $max(h_{2D\_obs}, h_{dubins})$ is the combination of the 2D Dijkstra's search and the Dubins heuristic obtained by taking the maximum of the two. Figure 6 shows an example of the total number of states that are expanded for a fixed start and goal location with the use of the above heuristics. Figure 6a is the obstacle filled test environment with start locations, goal locations and the planner output. To visualize the number of states expanded we assign each grid location (x, y) of the map with a color. Red indicates the most expansions, 16 in this case, while dark blue indicates no state expansions. Using this scheme Figure 6b, Figure 6c and Figure 6d show the results of using different heuristics. It can be seen that the combination of 2D Dijkstra's search and Dubins heuristics has the least number of states expanded.

IV. Theoretical Properties

We show that the algorithm is optimal in that it finds paths that go through a maximum number of waypoints and minimize the path costs. We divide the proof into two theorems. First we prove that given a single robot an optimal path can be found that maximizes the number of waypoints visited and minimizes path cost. Using this theorem we then prove that the multi-robot algorithm is optimal.

Theorem 1: *Given a set of motion primitives, a path found through the graph G_M visits as many waypoints as possible within the alloted time T_{MAX} while minimizing cost of the path.*

Let $\bar{G}_M \subset G_M$ be the mid level graph that contains only the states through which the goal can be reached within the time bound T_{MAX}. Let $\hat{\pi}_{\bar{G}_M}$ be the solution obtained by the planner with the cost $\sum_{i=1}^{\hat{k}+1} c(n_{i-1}, n_i)$ where $\hat{k}$ is the number of waypoints covered in this solution.

We define,
$c(\hat{\pi}_{\bar{G}_M}) = \sum_{i=1}^{\hat{k}+1} c(n_{i-1}, n_i) + (M - \hat{k}) \times P_M$, where $n \in V(G_M)$, $n_0 = n_{start}, n_{\hat{k}+1} = n_{goal}$, M = Total number of waypoints and $P_M = T_{MAX} + 1$.

Lemma 1: *All edges $\in E(\bar{G}_M)$ are optimal w.r.t the discretization of state space and action space since they are obtained from the low-level optimal graph search except for the edges connecting into goal states which have costs equal to the cost of least-cost path plus the penalty.*

Lemma 2: *As we run an optimal $A^\star$ search on the pruned graph $\bar{G}_M$, $\hat{\pi}_{\bar{G}_M}$ is the least cost path.*

We need to Prove:

1) $\hat{k}$ is the maximum number of waypoints that can be covered.
2) $\forall \pi_{\bar{G}_M}$ with fixed number of waypoints $\hat{k}$, $\sum_{i=1}^{\hat{k}+1} c(n_{i-1}, n_i)$ is the minimum cost.

Proof:

1) We prove by contradiction. Assume there is a path $\tilde{\pi}_{\bar{G}_M}$ that goes from start to goal whose time does not exceed T_{MAX} and has a larger number of waypoints $\tilde{k} > \hat{k}$.

$c(\hat{\pi}_{\bar{G}_M}) \leq c(\tilde{\pi}_{\bar{G}_M})$ from Lemma 2.

$\Rightarrow \sum_{i=1}^{\hat{k}+1} c(n_{i-1}, n_i) + (M - \hat{k}) \times P_M \leq \sum_{i=1}^{\tilde{k}+1} c(n_{i-1}, n_i) + (M - \tilde{k}) \times P_M$

$\Rightarrow (\tilde{k} - \hat{k}) \times (T_{MAX} + 1) \leq \sum_{i=1}^{\tilde{k}+1} c(n_{i-1}, n_i) - \sum_{i=1}^{\hat{k}+1} c(n_{i-1}, n_i)$

Any path cost in G_M is bounded from above by T_{MAX}.

$\Rightarrow (\tilde{k} - \hat{k}) \times (T_{MAX} + 1) \leq T_{MAX}$

This leads to a contradiction since $\tilde{k} > \hat{k}$. Thus, the maximum number of waypoints that can be covered is $\hat{k}$.

2) Assume another solution $\dot{\pi}_{\bar{G}_M} \in \bar{G}_M$ that has same number of waypoints, $\dot{k} = \hat{k}$.

$c(\hat{\pi}_{\bar{G}_M}) \leq c(\dot{\pi}_{\bar{G}_M})$ from Lemma 2.

$\Rightarrow \sum_{i=1}^{\hat{k}+1} c(n_{i-1}, n_i) + (M - \hat{k}) \times P_M \leq \sum_{i=1}^{\dot{k}+1} c(n_{i-1}, n_i) + (M - \dot{k}) \times P_M$

$\Rightarrow \sum_{i=1}^{\hat{k}+1} c(n_{i-1}, n_i) \leq \sum_{i=1}^{\dot{k}+1} c(n_{i-1}, n_i)$ since $\dot{k} = \hat{k}$.

Hence, $\sum_{i=1}^{\hat{k}+1} c(n_{i-1}, n_i)$ is minimum $\forall \pi_{\bar{G}_M}$ with fixed number of waypoints $\hat{k}$.

Theorem 2: *The path found through G_T has the minimum total number of unvisited waypoints and total path costs across all robots.*

Let $\bar{G}_T \subset G_T$ be the top level graph that contains only the states through which the goal can be reached within the time bound T_{MAX_i} for each robot. Let $\hat{\pi}_{\bar{G}_T}$ be the solution obtained by the planner and $\hat{k}$ be the number of unvisited waypoints in this solution.

We define,
$c(\hat{\pi}_{\bar{G}_T}) = \sum_{i=1}^{N} c(\pi_i) + \hat{k} \times P_T$, where $P_T = \sum_{i=1}^{N} T_{MAX_i} + 1$.

Lemma 1: *All edges $\in E(\bar{G}_T)$ are optimal since they are obtained from the mid-level optimal graph search except the edges connecting into goal states which have costs equal to the cost of least-cost path plus the penalty. This is proved in Theorem 1.*

Lemma 2: *As we run an optimal $A^\star$ search on the pruned graph $\bar{G}_T$, $\hat{\pi}_{\bar{G}_T}$ is the least cost path*

We need to Prove:

1) $\hat{k}$ is the minimum number of unvisited waypoints.
2) $\forall \pi_{\bar{G}_T}$ with fixed number of unvisited waypoints $\hat{k}$, $\sum_{i=1}^{N} c(\bar{\pi}_i)$ is the minimum cost, subject to $c(\bar{\pi}_i) \leq T_{MAX_i} \forall \pi_i$.

Proof:

1) We prove by contradiction. Assume a solution that has a less number of unvisited waypoints. Consider a path $\tilde{\pi}_{\bar{G}_T}$ such that it has less unvisited waypoints where $\tilde{k} < \hat{k}$.

$c(\hat{\pi}_{\bar{G}_T}) \leq c(\tilde{\pi}_{\bar{G}_T})$ from Lemma 2.

$\Rightarrow \sum_{i=1}^{N} c(\hat{\pi}_i) + \hat{k} \times P_T \leq \sum_{i=1}^{N} c(\tilde{\pi}_i) + \tilde{k} \times P_T$

$\Rightarrow (\hat{k} - \tilde{k}) \times P_T \leq \sum_{i=1}^{N} c(\tilde{\pi}_i) - \sum_{i=1}^{N} c(\hat{\pi}_i)$

$\Rightarrow (\hat{k} - \tilde{k}) \times (\sum_{i=1}^{N} T_{MAX_i} + 1) \leq \sum_{i=1}^{N} T_{MAX_i}$

This leads to a contradiction since $\tilde{k} < \hat{k}$. Thus, the minimum number of unvisited waypoints is $\hat{k}$.

2) Assume another solution $\dot{\pi}_{\bar{G}_T} \in \bar{G}_T$ that has same number of waypoints, $\dot{k} = \hat{k}$.

$c(\hat{\pi}_{\bar{G}_T}) \leq c(\dot{\pi}_{\bar{G}_T})$ from Lemma 2.

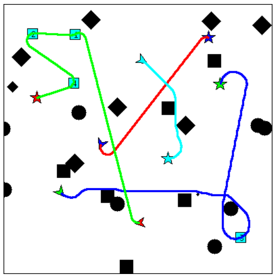
$\Rightarrow \sum_{i=1}^{N} c(\hat{\pi}_i) + \hat{k} \times P_T \leq \sum_{i=1}^{N} c(\dot{\pi}_i) + \dot{k} \times P_T$

$\Rightarrow \sum_{i=1}^{N} c(\hat{\pi}_i) \leq \sum_{i=1}^{N} c(\dot{\pi}_i)$ since $\dot{k} = \hat{k}$.

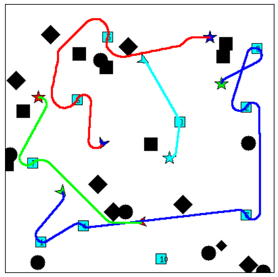
Hence, $\sum_{i=1}^{N} c(\hat{\pi}_i)$ is minimum $\forall \pi_{\bar{G}_M}$ with fixed number of waypoints $\hat{k}$.

V. Experimental Analysis

In our experiments we use fixed-wing UAVs. Model complexity can be greatly reduced by removing the vertical degree of freedom and working with constant altitude planar paths and further reduced by fixing airspeed to be constant. For constant speed applications, minimum turning radius is directly set by the upper bound on bank angle ($r_{min} = V^2/(g \times tan(\phi_{max}))$), where g is the gravitational constant and ϕ_{max} is the maximum angle of bank that can be achieved. The fundamental concept that must be captured by a planner is that turning is accomplished by rolling the vehicle for which there is an associated response lag. These physical constraints tend to reduce the fidelity of path planners which abstract dynamics away to work strictly with kinematic bounds on turning radius. As a consequence, motion primitives in the graph G_L are tailored to fit the gradual build-up of bank angle, which governs turning flight.



(a) Environment with four robots and four waypoints with 5% obstacle density



(b) Environment with four robots and ten waypoints with 5% obstacle density

Fig. 7: Simulation experiments environment examples

A. Simulation Experiments

For testing the algorithm in simulation an area of 10 square kilometers was discretized into 25 by 25 meter grids. Heading was discretized into 16 directions, thus altogether $400 \times 400 \times 16$ states in the low level graph G_L. Motion primitives were generated to achieve a turning radius of 270 meters for the simulated UAVs. Tests were conducted with varying number of robots and waypoints. Table I shows the planning time for our algorithm with three robots situated at fixed locations and different numbers of waypoints with no obstacles. In Table II the obstacle density is set to five percent. For a given number of waypoints, planning time was averaged over 20 randomly generated maps and waypoints locations were randomly chosen. All experiments were run on a PC with a 2.7 GHz Intel Core i7-2620M processor with 4 GB of RAM. Figure 7 shows two such randomly generated test scenarios.

B. Field Experiments

The Naval Postgraduate School (NPS) UAV lab has developed a Rapid Flight Test Prototyping System (RFTPS) to enable on-board integration of advanced control algorithms from concept to flight test. A new approach to path following and coordination was developed in [30]. For our tests we used two SIG Rascal UAVs shown in Figure 8c.

These algorithms were flight tested at Camp Roberts, CA where NPS conducts UAV experiments. Further details can be found in [31]. A typical scenario involving two UAVs and four waypoints is illustrated in Figure 9. The site is roughly 6x4 sq-km. The red polygon outlines the boundary

TABLE I: Planning time (seconds): varying robots and varying number of waypoints with no obstacles

# UAV \ # Waypoint	5	7	9	11
3	0.1	0.85	2.11	9.14
5	0.98	6.40	12.43	83.40

TABLE II: Planning time (seconds): varying robots and varying number of waypoints with five percent obstacle density

# UAV \ # Waypoint	5	7	9	11
3	38.63	65.15	84.79	136.70
5	60.48	74.60	108.23	182.13

(a) Mobile Ground Control Station

(b) Interior of GCS

(c) Two SIG Rascal UAVs

Fig. 8: Experimental Setup for field tests

of cleared airspace at the Camp Roberts range. The paths generated by the planner are shown in dark blue and green. For the UAVs, motion primitives were generated to achieve a turning radius of 270 meters which is twice the r_{min}. This allocates half of the aircraft turning authority to the planner and the other half to disturbance rejection. Results of these experiments validated that paths generated by the planner are feasible. Even in high winds the average error between the commanded path and the tracked path was not more than 50m which is equivalent to two seconds at the planned velocity. The supplemental video presents one of the experimental run at Camp Roberts.

VI. Conclusion

The Orienteering Problem and the Team Orienteering Problem have been extensively studied in the Operations Research community where they are usually formulated as an optimization problem. In this paper we studied the OP and the TOP with respect to mobile robots with dynamic constraints, particularly the bounds on turning radius and its rate of change. While others have investigated orienteering by assuming the costs between the locations to be given, to our knowledge this is the first attempt to combine orienteering with the dynamic constraints of the robots. We formulate the OP and the TOP as a graph search problem and address the

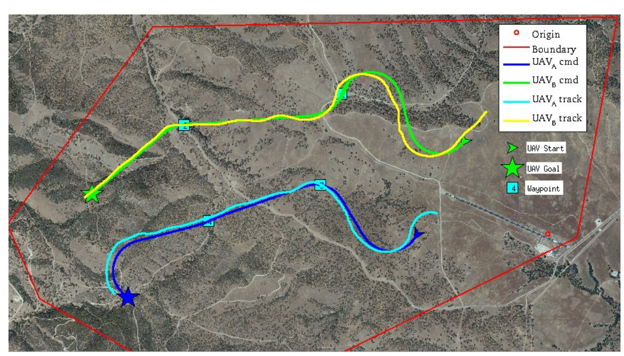


Fig. 9: Plot showing plan generated for two UAVs and four waypoints and the tracked positions of two UAVs

dynamic constraints of the robots. We also explored the use of Dubins Curves as heuristics for graph search algorithms. A combination of Dubins Curves and 2D Dijkstra's search as heuristics for the $A^\star$ search of a 3D graph gave the best results. Further, we solved the OP and TOP using the *Single Robot Opportunistic Path Refinement* and *Multi Robot Opportunistic Path Refinement* algorithm respectively. We established in theory that our algorithm is optimal *i.e.* the robots cover a maximum number of waypoints in the minimum amount of time with respect to the discretization of state space and action space. We validated the outputs of our algorithm by testing it in simulation and flight tests using SIG Rascal UAVs.

In future work we would like to bring in methods that trade off optimality for runtime with provable bounds on suboptimality in order to scale to large teams of UAVs and 20-100 waypoints. Also, in many cases the paths generated for the individual robots must be replanned to accommodate waypoints added in the mission while executing. This requires replanning in real time and during flight. Currently, the algorithm is centralized and we are exploring the search direction that will allow decentralization.

VII. Acknowledgments

We would like to acknowledge the Joint Interagency Field Exploration (JIFX) organizers at the Naval Postgraduate School who enabled us to validate these algorithms in a flight test setting at their McMillan Airfield, Camp Roberts, CA facility. Furthermore, we gratefully acknowledge support from ONR grants N00014-09-1-1031 and 10936907.

References

[1] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. In Press, Corrected Proof, 2010.

[2] B. Golden, L. Levy, and R. Vohra, "The Orienteering Problem," *Naval Research Logistics*, vol. 34, pp. 307–318, 1987.

[3] G. Laporte and S. Martello, "The selective travelling salesman problem," *Discrete Appl. Math.*, vol. 26, no. 2-3, pp. 193–207, 1990.

[4] M. K. R. Ramesh, Y. Yong Seok, "An optimal algorithm for the orienteering tour problem," *ORSA Journal on Computing*, vol. 4, pp. 155–165, 1992.

[5] M. Fischetti, J. J. S. Gonzalez, and P. Toth, "Solving the orienteering problem through branch-and-cut," *INFORMS J. on Computing*, vol. 10, no. 2, pp. 133–148, 1998.

[6] F. S. M. Gendreau, G. Laporte, "A Branch-and-Cut algorithm for the undirected selective traveling salesman problem," *Networks*, vol. 32, pp. 263–273, 1998.

[7] T. Tsiligirides, "Heuristic Methods Applied to Orienteering," *Journal of the Operational Research Society*, vol. 35, pp. 797–809, 1984.

[8] L. B.L. Golden, Q. Wang, "A Multifaceted Heuristic for the Orienteering Problem," *Naval Research Logistics*, vol. 35, p. 359366, 1988.

[9] I.-M. Chao, B. L. Golden, and E. A. Wasil, "A fast and effective heuristic for the orienteering problem," *European Journal of Operational Research*, vol. 88, no. 3, pp. 475–489, February 1996.

[10] C. P. Keller, "Algorithms to solve the orienteering problem: A comparison," *European Journal of Operational Research*, vol. 41, no. 2, pp. 224 – 231, 1989.

[11] R. Ramesh and K. M. Brown, "An efficient four-phase heuristic for the generalized orienteering problem," *Computers and Operations Research*, vol. 18, no. 2, pp. 151–165, 1991.

[12] B. L. G. Qiwen Wang, Xiaoyun Sun and J. Jia, "Using artificial neural networks to solve the orienteering problem," *Annals of Operations Research*, vol. 61, no. 1, pp. 111–120, 1995.

[13] M. Gendreau, G. Laporte, and F. Semet, "A tabu search heuristic for the undirected selective travelling salesman problem," *European Journal of Operational Research*, vol. 106, no. 2-3, pp. 539 – 545, 1998.

[14] M. F. Tasgetiren, "A genetic algorithm with an adaptive penalty function for the orienteering problem," *Journal of Economic and Social Research*, vol. 4, no. 2, pp. 1–26, 2001.

[15] Y.-C. Liang, S. Kulturel-Konak, and A. E. Smith, "Meta heuristics for the orienteering problem," in *Proceedings of the 2002 Congress on Evolutionary Computation CEC2002*, D. B. Fogel, M. A. El-Sharkawi, X. Yao, G. Greenwood, H. Iba, P. Marrow, and M. Shackleton, Eds. IEEE Press, 2002, pp. 384–389.

[16] S. E. Butt and T. M. Cavalier, "A heuristic for the multiple tour maximum collection problem," *Computers and Operations Research*, vol. 21, no. 1, pp. 101–111, 1994.

[17] S. Boussier, D. Feillet, and M. Gendreau, "An exact algorithm for team orienteering problems," *4OR: A Quarterly Journal of Operations Research*, vol. 5, pp. 211–230, 2007.

[18] I.-M. Chao, B. L. Golden, and E. A. Wasil, "The team orienteering problem," *European Journal of Operational Research*, vol. 88, no. 3, pp. 464–474, February 1996.

[19] H. Tang and E. Miller-Hooks, "A tabu search heuristic for the team orienteering problem," *Comput. Oper. Res.*, vol. 32, no. 6, pp. 1379–1407, 2005.

[20] C. Archetti, A. Hertz, and M. Speranza, "Metaheuristics for the team orienteering problem," *Journal of Heuristics*, vol. 13, pp. 49–76, 2007.

[21] L. Ke, C. Archetti, and Z. Feng, "Ants can solve the team orienteering problem," *Computers & Industrial Engineering*, vol. 54, no. 3, pp. 648–665, 2008.

[22] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. V. Oudheusden, "A guided local search metaheuristic for the team orienteering problem," *European Journal of Operational Research*, vol. 196, no. 1, pp. 118 – 127, 2009.

[23] H. Bouly, D.-C. Dang, and A. Moukrim, "A memetic algorithm for the team orienteering problem," *4OR: A Quarterly Journal of Operations Research*, vol. 8, pp. 49–70, 2010.

[24] W. Souffriau, P. Vansteenwegen, G. Vanden Berghe, and D. Van Oudheusden, "A path relinking approach for the team orienteering problem," *Comput. Oper. Res.*, vol. 37, no. 11, pp. 1853–1859, 2010.

[25] S. Bhattacharya, M. Likhachev, and V. Kumar, "Multi-agent Path Planning with Multiple Tasks and Distance Constraints," *IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[26] M. Likhachev and D. Ferguson, "Planning Long Dynamically-Feasible Maneuvers For Autonomous Vehicles," *International Journal of Robotics Research (IJRR)*, 2009.

[27] M. Pivtoraiko and A. Kelly, "Generating near minimal spanning control sets for constrained motion planning in discrete state spaces," in *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '05)*, August 2005, pp. 3231 – 3237.

[28] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems, Science, and Cybernetics*, vol. SSC-4, no. 2, pp. 100–107, 1968.

[29] 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.

[30] E. Xargay, V. Dobrokhodov, I. Kaminer, A. Pascoal, N. Hovakimyan, and C. Cao, "Time-critical cooperative control of multiple autonomous vehicles: Robust distributed strategies for path-following control and time-coordination over dynamic communications networks," *Control Systems, IEEE*, vol. 32, no. 5, pp. 49 –73, oct. 2012.

[31] J. Keller, D. Thakur, V. Dobrokhodov, K. Jones, M. Pivtoraiko, J. Gallier, I. Kaminer, and V. Kumar, "A computationally efficient approach to trajectory management for coordinated aerial surveillance," *Unmanned Systems*, vol. 01, no. 01, pp. 59–74, 2013.