Provably Correct Persistent Surveillance for Unmanned Aerial Vehicles Subject to Charging Constraints

Kevin Leahy, Dingjiang Zhou, Cristian-Ioan Vasile, Konstantinos Oikonomopoulos, Mac Schwager, and Calin Belta

Boston University, Boston MA 02215

1 Motivation, Problem Statement, and Related Work

In this paper, we investigate the automatic deployment of multiple quadrotors under resource constraints. The relatively short battery life in many unmanned aerial vehicles (UAVs) presents a significant barrier to their use in complex, long term surveillance missions. Moreover, the use of multiple vehicles allows for more complex behavior and longer mission horizons, but further complicates the task of deploying those vehicles given limited flight time. We present an algorithm that generates a feedback controller for multiple quadrotors with charging constraints to meet a complex temporal logic specification. The algorithm comprises a three-part tool chain that first plans a high level routing schedule for the quadrotors, then generates a vector field control input for the quadrotors to accomplish the schedule, and finally controls the quadrotors' nonlinear dynamics to follow the vector field with a feedback controller. The performance of the complete system, with its three interacting parts, is investigated in 50 experimental runs using two quadrotors in a motion capture environment.

As a motivating example, we consider the environment shown in Fig. 1 consisting of three charging stations, three regions of interest, and two aerial vehicles. We assume vehicle battery life is 40 time units, and charging takes 120 time units, where time units are a generic unit that can be instantiated based on a particular implementation. Given this environment and these battery and charging constraints, we wish to satisfy the following mission specification infinitely often: "within 16 time units observe Region R3 for at least 3 time units; within 28 time units, observe Region R1 for at least 2 time units; and within 46 time units, observe Region R2 for at least 2 time units then within 8 time units observe Region R1 or Region R3 for at least 2 time units." We seek a method to generate a control policy ensuring that vehicles can be automatically deployed to successfully complete this mission in the specified environment.

Formally, the problem we seek to solve is: given an environment and a temporal logic mission specification with time deadlines that needs to be satisfied infinitely often, generate control policies for a team of quadrotors to complete the mission, while ensuring vehicles remain charged and collisions are avoided.

The solution to this problem requires the use of several sophisticated systems, whose interaction both at a theoretical level and an experimental level produces many unique challenges. Our approach is related to the Vehicle Routing Problem

(VRP) [1], which can be summarized as: given N identical vehicles at a depot and the distances among all sites and the depot, find a minimum distance tour for each vehicle such that it begins and ends at the depot and visits each site at least once. By placing time bounds on when each site must be visited, we obtain a version of VRP called Time Window VRP (VRPTW). Recent related work includes [2] where the authors propose a fragment of metric temporal logic, which restricts temporal operators to atomic propositions and their negation. Our approach has fewer restrictions, including allowing for a vehicle to visit a site multiple times during a tour if it is required. Multi-agent control for the VRPTW has also been considered without temporal logic constraints in [3][4].

2 Technical Approach

To capture the richness of the specification, we use bounded linear temporal logic (BLTL), a temporal logic with time bounds on each of its temporal operators. The mission specification presented above can be expressed as $\mathbb{G}\phi$, where ϕ is given in (1) as a BLTL formula and the $\mathbb{G}$ operator indicates that ϕ should be satisfied infinitely often.

$$\phi = \mathbf{F}^{\leq 16}\mathbf{G}^{3}\pi_3 \wedge \mathbf{F}^{\leq 28}\mathbf{G}^{2}\pi_1 \ \wedge \mathbf{F}^{\leq 46}\left(\mathbf{G}^{2}\pi_2 \wedge \mathbf{F}^{\leq 10}\mathbf{G}^{2}\left(\pi_1 \vee \pi_3\right)\right) \qquad (1)$$

In (1), $\wedge$ and $\vee$ are the usual Boolean operators indicating conjunction and disjunction, while $\mathbf{F}$ and $\mathbf{G}$ are the temporal operators"eventually" and "always," respectively. Superscripts on the temporal operators are time bounds on those operators. Each π_i is the atomic proposition that is true when Region Ri is being observed. Flight through the environment generates a word over Π, the set of the atomic propositions, which can be checked against (1) for satisfaction of the specification.

Finding a control policy that satisfies (1) necessitates representing the environment as a weighted transition system, where the charging stations and regions of interest are the states and the transitions are possible paths among the states. The weights on the transitions represent travel times along those paths.

To generate the transition system, the environment is first partitioned into cubes. Paths between all pairs of regions of interest are found as sequences of these cubes. The paths are constrained such that quadrotors fly to a fixed height from the charging stations and perform all observations from that fixed altitude. From these paths, a vector field is generated to ensure each sequence of cubes is followed. The vector field everywhere inside a given cube is a convex combination of the vectors at its vertices [5]. Such a vector field can be used to keep the vehicle from leaving the cube (stay-in-cell) or to force it to leave through a given facet (control-to-facet), as displayed in Fig. 2. The weights on the transition system are the upper bounds on travel time along each sequence of cubes, which can be calculated using methods previously explored [6]. Hovering over a region or charging are modeled as self-loop transisitons of weight 1.

Once the transition system is created, the charging and battery constraints are converted to finite state automata (FSA). A series of product automata are created from the charging FSA; specification (1), encoded as an FSA;and the transition system. Djikstra's algorithm is used on the final product automaton to find satisfying minimum loop-time control strategies, for which the vector fields have already been generated. Only mutually exclusive operation of quadrotors is considered in this framework and thus collision avoidance is conservatively guaranteed.

Following the vector field is accomplished using a controller that exploits the fact that quadrotor dynamics are known to be differentially flat [7]. The controller calculates a control input based on a flat output—the quadrotor position and yaw angle—and its derivatives, which can be calculated analytically from the vector field.

3 Results

The partitioned environment (Figs. 1 & 5) consists of 385 cubes each with edge length 0.36m. Control policies for $\mathbb{G}\phi$ were calculated over the transition system displayed in Fig. 1. The computation time, excluding encoding of (1), was 301.7 seconds, and the final product automaton had 579,514 nodes and 2,079,208 edges. No solutions were found for quadrotors starting on Chargers C2 and C3, but all other combinations of starting positions yielded solutions.

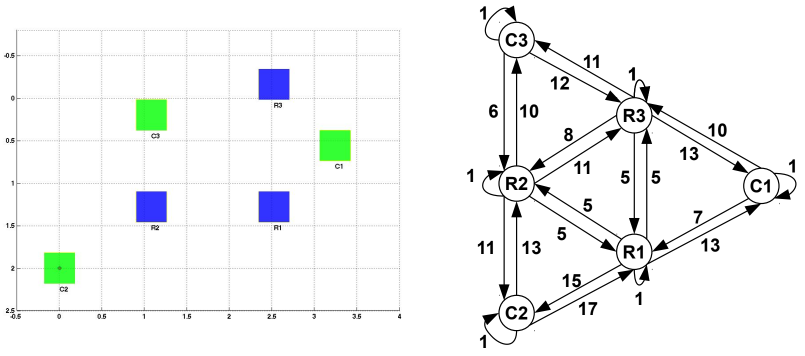

Fig. 1. Partitioned environment viewed from above and transition system. Green squares are charging stations, while blue squares are regions of interest. States in the transition system are charging stations and regions of interest. Weights on transitions are based on calculated time bounds.

4 Experiments

Experiments were performed in the Boston University Aerial Robotics Arena. The Arena consists of a flight space with IR cameras to track reflective markers

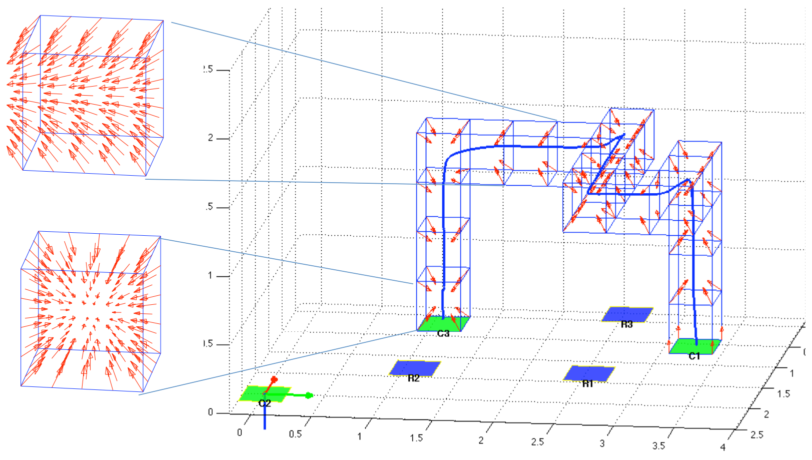

Fig. 2. Vector field detail and quadrotor flight data displayed in vector field. The cube at the top left shows a control-to-facet vector field, and the cube at the bottom left shows a stay-in-cell vector field. One of these two kinds of fields is given to the quadrotor in each cell along its path to guide it through the desired trajectory.

on the quadrotors using an OptiTrack system. This system allows for real-time localization of the quadrotors during experiments. Two K500 quadrotors from KMel robotics were used to execute the control policies described in Sec. 3.

Charging stations (Fig. 3) were designed and built at Boston University for automatic docking and charging of quadrotors. These platforms allow one vehicle to land and another to take off, ensuring continuous monitoring in situations where one vehicle may not be able to satisfy a persistent monitoring mission specification on its own. The charging stations are made of laser cut acrylic parts connected with PLA plastic 3D printed parts. The electronics of the station consist of the Hyperion EOS0720i Net3AD charger, modified to enable control by MATLAB. The charging pads are 410 stainless steel. To secure a robust connection with the stainless steel pads of the charging station, the quadrotors are equipped with stainless steel contacts mounted on springs with magnets. The platform is entirely controlled by MATLAB via USB connection, allowing for the detection of the presence of a quadrotor, real-time monitoring of battery and charging status, and control of the charging parameters including battery type, capacity, and charging rate. The maximum charging rate that can be achieved is 8 Amperes.

Figure 4 shows the results of a flight by two quadrotors. Seconds were used as the time units for these experiments so flights could be rapidly performed and analyzed. The flights consist of two loops that each satisfy ϕ. Since ϕ can be satisfied repeatedly, these flights can satisfy the mission specification, $\mathbb{G}\phi$. In the first loop, Quadrotor 1 takes off first and Quadrotor 2 completes the loop, while in the second loop the flight order is reversed. The second loop of the flight resulted in the quadrotors landing on the same charging stations from which they took off, and any subsequent loops would be identical to this second loop.

Figure 4 shows that the specification was satisfied for both loops in the flight. Region R1 was visited in 5.8 seconds in Loop 1 and 7.5 seconds in Loop 2, ahead of the 28 second deadline. Likewise, Region R3 was visited in 12.4 and 12.6 seconds ahead of the 16 second deadline. In the second portion of each loop, Region R2 was visited in 34.0 and 30.3 seconds with a deadline of 46 seconds, and Region R1 was visited within the 8 second deadline after each visit to Region R2.

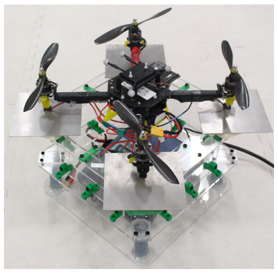
Fig. 3. Quadrotor resting on charging station.

The two-loop flight described above was performed 50 times, and both quadrotors were consistent in their flight times. The standard deviation in the length of each portion of the flight time was on the order of $0.1s$. Despite this consistency, the time bound on flying from Charger C1 to Region R1 was violated by the second quadrotor in each flight, while not being violated by the first quadrotor.

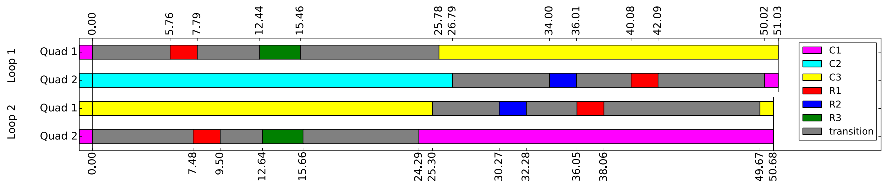

Fig. 4. Timeline of quadrotor flights for two loops. The first two rows display the first loop, with Quadrotor 1 flying before Quadrotor 2. The next two rows show the second loop, with Quadrotor 2 flying first.

5 Main Experimental Insights

The implementation of the persistent surveillance framework required three systems to be integrated together: the BLTL control policy algorithm, the vector field, and the quadrotor controller. Inevitably, the limitations to a theoretical framework appear at the interface of such systems. For example, the use of multiple vehicles required tuning the controllers quite differently to ensure that the vector field was followed, even though the vehicles are of the same make and model. Regardless of any such complications, because a conservative approach was used, such as using upper bounds on travel time rather than expected travel time, the system met the specifications reliably and predictably. These experiments establish a framework that can be extended to larger groups of vehicles flying simultaneously, more complex distributed mission tasks, and longer experimental mission horizons.

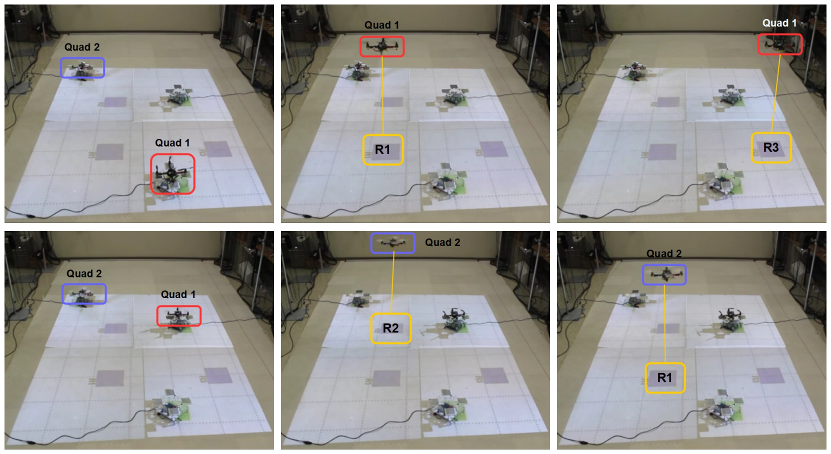

Fig. 5. Screencaps of the first flight loop.

The main insight gleaned from this experiment is how to automate a complex persistent surveillance mission specified as a temporal logic formula. The methodology explained herein allows for rapid experimentation following theoretic work using temporal logics. By using the environment partition and transition system generation with time bounds, minimal human input is required to establish an experimental framework for simulating and executing missions. Further, the inclusion of charging stations, whose performance can be modeled using automata, allows for long-term, truly persisent missions involving multiple vehicles not only to be modeled, but to actually be performed in the lab.

References

1. G.B. Dantzig and J.H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.
2. S. Karaman and E. Frazzoli. Vehicle routing problem with metric temporal logic specifications. In *Proc. of the IEEE Conference on Decision and Control (CDC08)*, pages 3953–3958. IEEE, 2008.
3. N. Michael, E. Stump, and K. Mohta. Persistent surveillance with a team of mavs. In *Proc. of the International Conference on Intelligent Robots and Systems (IROS 11)*, pages 2708–2714. IEEE, 2011.
4. E. Stump and N. Michael. Multi-robot persistent surveillance planning as a vehicle routing problem. In *Proc. of the IEEE Conference on Automation Science and Engineering (CASE)*, pages 569–575. IEEE, 2011.
5. C. Belta and L.C.G.J.M. Habets. Controlling a class of nonlinear systems on rectangles. *IEEE Transactions on Automatic Control*, 51(11):1749–1759, 2006.
6. E. Aydin Gol and C. Belta. Time-constrained temporal logic control of multi-affine systems. *Nonlinear Analysis: Hybrid Systems*, 10:21–33, 2013.
7. D. Zhou and M. Schwager. Vector field following for quadrotors using differential flatness. In *Proc. of the International Conference on Robotics and Automation (ICRA)*, May 2014.