Decentralized Perimeter Surveillance Using a Team of UAVs

Derek Kingston, Randal Beard, and Ryan Holt

***Abstract*—This paper poses the cooperative perimeter surveillance problem and offers a decentralized solution that accounts for perimeter growth (expanding or contracting) and insertion/deletion of team members. By identifying and sharing the critical coordination information and by exploiting the known communication topology, only a small communication range is required for accurate performance. Simulation and hardware results are presented that demonstrate the applicability of the solution.**

***Index Terms*—cooperative control, perimeter surveillance, decentralized algorithms, small UAVs, coordination variables**

I. Introduction

PERIMETER surveillance algorithms form the basis for effective monitoring in a number of applications including monitoring oil spills [1], contaminant clouds [2], algae bloom [3], forest fires [4], [5], and border security [6], [7].

Our aim is to develop algorithms that operate on small UAVs which offer some distinct advantages over larger UAVs. Small UAVs can be man portable and hand launchable, removing the need for traditional runways and allowing teams to be easily and rapidly deployed even in rough terrain. As a relatively inexpensive platform, large numbers of small UAVs can be deployed to increase the rate at which information is gathered. The use of small UAVs creates unique requirements for the cooperation algorithms that control teams of vehicles. Algorithms must be robust to loss of agents since small UAVs are more susceptible to weather conditions and are more fragile than larger UAVs. The communication packages onboard small UAVs are often low-power, requiring that communication constraints be explicitly addressed in their cooperation strategies. Finally, the computational burden should remain constant regardless of team size, i.e. the cooperation algorithm should scale well for large teams. Since a cooperation algorithm that is robust, addresses communication constraints, and is scalable to large teams will work on both large and small UAVs, we focus our efforts on developing such algorithms.

Derek Kingston is with the Air Vehicles Directorate, Air Force Research Laboratory, Wright-Patterson AFB, OH 45433 (corresponding author email: derek.kingston@wpafb.af.mil)

Randal Beard is a professor in the Electrical and Computer Engineering Department of Brigham Young University, Provo, UT 84602 (email: beard@byu.edu)

Ryan Holt is with MIT Lincoln Laboratory, Lexington, MA 02421 (email: rholt@ll.mit.edu)

This research was supported by NASA under STTR contract No. NNA04AA19C to Scientific Systems Company, Inc (SSCI) and Brigham Young University (BYU), by the National Science Foundation under Information Technology Research Grant CCR-0313056, and by the United States Air Force under AFOSR Award number FA9550-04-1-0209.

We are particularly interested in monitoring borders that are of unknown shape and size and possibly changing in time like those that would be encountered in monitoring a forest fire or chemical spill. Additionally, we do not exclude large borders where communication range will limit the possible interaction of the team. We will assume that UAV agents have the proper sensor suite to detect changes in the perimeter and track the edge of the perimeter. We will not focus on the necessary sensor technology to do this, but rather on the algorithms that will allow a team of agents to monitor a perimeter in a decentralized fashion. Perimeter surveillance using multiple UAVs has the advantage of operating in a wide variety of situations like changing perimeters (spill monitoring, forest fire surveillance) or very large perimeters (border patrol).

A number of researchers have investigated similar problems of monitoring/tracking changing perimeters with autonomous vehicles. The MDARS project [8] is a joint effort between the US Army and Navy that networks multiple ground robots to cooperatively monitor a fixed perimeter near critical storage facilities. A team of robots are equipped with coarse obstacle detection sensors and a high precision narrow field of view sensor to find and track objects that have breached the perimeter [7]. The entire team of vehicles communicates to a central location where sensor data is fused and waypoint commands are issued [6]. Our work differs from MDARS in that we do not require team agents to be in constant communication with a centralized controller; rather, agents are frequently outside of the communication range of the other team members and must monitor the perimeter in a decentralized manner. Information gathered by the team is then carried to a base location where the state of the perimeter is displayed and human operators make decisions.

Teams of autonomous underwater vehicles have been proposed as a way to track algae bloom and oil spills. In Ref. [9], Bertozzi et al. present an algorithm for monitoring a perimeter with multiple vehicles where each vehicle is equipped with a concentration sensor. When the sensor detects the presence of the chemical, the vehicle turns in one direction; in the absence of chemical detection, the vehicle turns in the opposite direction. In this way, an agent weaves around the perimeter of the spill while communicating the perimeter crossing points to completely survey the perimeter. A simple spacing law adjusts the speed of the vehicles to allow the team to spread out along the perimeter. The algorithm has been shown to work in hardware testbed experiments with virtual perimeters [10].

Clark and Fierro propose a similar method for oil spill perimeter tracking using multiple vehicles [1]. A fleet of vehicles is deployed and will search the region and communicate

to team members when the perimeter is located. Agents will approach the perimeter and begin to track it in a predetermined direction. Spacing of the vehicles is accomplished by adjusting linear velocity. Hardware experiments using a camera sensor on wheeled robots is shown to validate the algorithm. In both this approach and the one proposed by Bertozzi et al. [9], neither the efficiency nor the convergence of the algorithms are shown analytically. In addition, the problem of limited communication range is not addressed.

Susca, Martinez and Bullo address the issue of approximating a changing border with a set of interpolation points [11]. As the team agents traverse the perimeter, they update the points that describe the perimeter to best fit a polygon to the shape of the perimeter. Their algorithm is shown to converge and relies only on communication between immediate neighbors.

In this paper, we will present an algorithm for perimeter surveillance that: (1) is completely decentralized, (2) is provably convergent to the optimal behavior in finite-time, (3) explicitly accounts for communication range limitations, and (4) allows for changing perimeters. The primary advantages to a decentralized approach are scalability and inherent system robustness. Since agents only make decisions based on neighbor interactions, the required communication bandwidth and computation is fixed irrespective of the total number of agents on the team. Decentralization is inherently robust since each agent makes decisions with its available information without a need to receive directions from a central location. This eliminates single points of failure and allows a system to adapt naturally to changes in team size. Agents can be inserted and deleted from the team at any time and the system will adjust since each agent will maneuver to find its new neighbors. This allows agents to leave the team for high priority tasks such as following a perimeter breach, or in case of an accident or the need to refuel.

In addition to being fully decentralized, our approach is optimal at steady-state and has finite-time convergence. Additionally, our approach requires very little communication bandwidth and accounts for UAV kinematic constraints. The algorithm is limited to constant velocity vehicles that travel along the border, and due to its decentralized nature, any global information that may be available is not exploited. For missions where robustness is valued more than efficiency, our approach is a natural fit. Since it guarantees optimality in steady-state and finite-time convergence, only missions that have strict efficiency requirements would not be well-suited to the approach.

This paper is based upon a previous publication by the authors [12], but extends it in a number of ways. In particular, the algorithms are more precise in definition and time bounds for tracking changing perimeters are proven. Additionally, hardware results have been added to support the simulation results of [12].

The perimeter surveillance problem is posed in Sections II and III. Section IV presents our solution using a coordination variable [13] approach. The method is extended to changing perimeters in Section V and to account for constrained UAV turning radius in Section VI. Simulation and hardware results are presented in Sections VII and VIII. Finally, Section IX gives our conclusions.

II. Problem Formulation

The objective of the cooperative perimeter surveillance problem is to cooperatively gather information about the state of the perimeter and to transmit that data to a central base station with as little delay and at the highest rate possible. There are a number of factors that complicate the perimeter surveillance problem including:

1) Perimeter topology
2) Communication constraints
3) Team logistics
4) UAV capability.

Perimeter Topology. A perimeter may be static, such as a well-defined border, or changing in time, such as a chemical spill or forest fire. A perimeter can be composed of a web of segments and nodes that must be monitored, such as a set of city streets or a network of paths in the mountains, although we do require the graph representing the perimeter to be strongly connected. An area surveillance problem can sometimes be posed as a perimeter surveillance problem by constructing a path that covers the area using, for example, a Zamboni pattern, and then monitoring that path as a perimeter. The perimeter location need not be known *a priori*, but when this is the case we assume that the UAVs have the sensor capacity to detect and follow the perimeter autonomously.

Communication Constraints. Small, inexpensive UAVs often have limited communication bandwidth and short communication range. In scenarios where the perimeter is very large or the terrain causes line-of-sight problems, agents may frequently be out of communication range of the base station and neighboring UAVs. Additionally, the gathered data may require significant time to transmit when a UAV is in communication range of its neighbors (e.g. complete video footage).

Team Logistics. UAVs have limited flight time and must be periodically refueled. In many cases, a UAV may be re-tasked to investigate a perimeter breach. Hardware failures and hazardous flying conditions may unexpectedly remove a UAV from involvement. A perimeter surveillance solution should be robust to failures and allow for interruptions like reassignment and refueling.

UAV Capability. The maneuverability of the UAV agents also effects the monitoring of a perimeter. We assume that the UAVs are equipped with GPS and autopilot similar to the one described in Ref. [14]. The autopilot maintains constant altitude and each UAV on the team is given a unique altitude assignment. Neglecting wind, the kinematic equations of motion for a single UAV can be written as

$$\dot{p}_N = V\cos\psi \tag{1}$$

$$\dot{p}_E = V\sin\psi \tag{2}$$

$$\dot{\psi} = \omega, \quad |\omega| < \omega_{\max} \tag{3}$$

where $\mathbf{p} = (p_N, p_E)^T$ is the inertial position of the UAV, ψ the heading, V the airspeed, and ω the heading rate. The

kinematics are augmented by a constraint on ω to capture the curvature constraint of UAV motion. This model of UAV motion is known as the Dubin's model and has shown great value for design of UAV systems as it captures the essential navigational kinematics of UAV motion while at the same time being of low enough order to allow tractable analysis [15].

Developing a perimeter surveillance algorithm that accounts for these complications and efficiently gathers data about the perimeter state is not trivial. We reduce the general problem to a more manageable, but still applicable, problem and present the team behavior that efficiently solves that problem in Section III. Section IV then introduces and proves the convergence of an algorithm for reaching the desired behavior while accounting for limited communication range. Section VI discusses the application of the simple algorithm to the kinematics given in Equations (1)–(3).

III. Linear Perimeter Surveillance

We reduce the general perimeter surveillance problem of Section II to the linear surveillance problem by assuming that the perimeter to be monitored is homeomorphic to a line and can therefore be represented as a single path between two points. This assumption eliminates perimeters that are circular or that are connected in a web-like structure. However, an arbitrary connected perimeter can be reduced to a linear perimeter by constructing a single tour that traverses all segments of the original perimeter. In practice, a surveillance mission will have a base of operations where information about the perimeter is analyzed by human operators and team agents are refueled and relaunched. Circular perimeters can be treated as linear perimeters with both endpoints at the base of operations.

A linear perimeter imposes a natural order to the team where each agent has at most two immediate neighbors along the perimeter. By requiring that neighbors physically meet to transmit information, any size of communication range is allowed. In practice, the sensor footprint limitations will require UAVs to physically meet their neighbors regardless of whether they can communicate at larger distances. Therefore we assume that UAVs must meet to exchange information. Agent meeting times can be extended by loiter patterns to facilitate the transmission of large amounts of data. Loss or reassignment of team agents are quickly noticed by the change in the neighborhood of affected agents.

Team planning is accomplished by considering agents as point masses that move at uniform constant velocity along the perimeter (see Figure 1). Corresponding UAV agents follow their reference points along the perimeter as described in Section VI. We assume that point agents can reverse direction instantaneously and that they always do so at the end of the perimeter. Communication between point agents is only allowed when they are "touching", i.e. when they occupy the same physical location. One way to visualize the problem is to imagine beads sliding along a wire, as in Figure 1.

Considering the agents to be point masses that travel at constant velocity, it may seem that posing the team as a first-order dynamical system would allow well developed tools

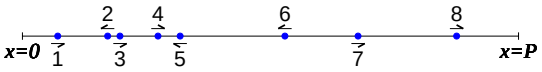

Fig. 1. Example scenario where 8 agents monitor a linear perimeter.

from control theory to be applied. However, due to the discrete nature of the input (only velocity direction is controlled) and the logic that determines this input, we found that a distributed systems approach similar to those used in the computer science field [16] enabled better analysis. Indeed, the following theorems regarding system convergence follow a logical progression based on the decision mechanism rather than a control systems approach.

The performance of a particular monitoring algorithm can be measured by the latency associated with information about points along the perimeter. Let P be the length of the perimeter and let the perimeter be defined as a line along the x-axis beginning at $x = 0$ and continuing until $x = P$. Since we assume that the point agents travel at uniform velocity V and data transmission only occurs when agents are in immediate physical proximity, the soonest information about point x_0 is available to a recipient at the base of operations ($x = 0$) is in x_0/V seconds. The minimum latency profile is obtained when an agent starts at the far end of the perimeter and travels to the base of operations, at which time it transmits all of the perimeter information.

Note that adding more agents cannot decrease the *latency* of the gathered information as seen at the base of operations since information can only travel as fast as a single agent. However, increasing the number of agents on the team increases the *refresh rate* of the perimeter state. Intuitively, spacing agents equally so that the refresh rate is constant will yield the most efficient method for perimeter monitoring. This configuration can be achieved by tasking each agent to travel to the end of the perimeter and then monitor the entire perimeter as it returns to the base while launching agents at $2P/N$ intervals where N is the number of agents on the team. As agents monitor the perimeter while traveling to the base of operations they pass agents traveling to the end of the perimeter to begin monitoring. These meetings occur at equally spaced intervals of length P/N. Rather than have agents traverse the entire perimeter equally spaced, each can be responsible for a segment of length P/N and pass the information it gathers to its neighbors, thus achieving the same overall latency profile and refresh rate.

Consider the behavior of a team of four agents as shown in Figure 2. The agents are uniformly distributed along the perimeter (Figure 2(a)) and each agent meets its neighbors at the end of its segments (Figures 2(b) and 2(c)). This oscillatory behavior of the agents requires that the team be synchronized not only in space (equally distributed), but also in time (meet neighbors at the end of segments).

By examining the behavior illustrated in Figure 2 it can be seen that gathered information travels to all other locations along the perimeter in the shortest time possible. This can be seen by noting that after two agents meet and exchange information, each will take this information at speed V to all

Fig. 2. Information exchange pattern that allows information about the state of the perimeter to be available at any point along the perimeter. a) Agents are uniformly spaced along the perimeter. b) Neighbors meet and exchange perimeter state information. c) Perimeter state is carried to all points along the perimeter.

other places along their respective segments. This information is passed to their respective neighbors who carry it further along the perimeter, again at speed V. Therefore, the information gathered when neighbors meet is carried to all other points along the perimeter at the highest possible speed.

Definition 1 (Low-Latency Exchange Configuration): Consider a team of N agents monitoring a linear perimeter of length P defined as a line along the x-axis from $x = 0$ to $x = P$. Order the agents from the left end of the perimeter as $1 \dots N$. Consider the following two sets of team locations on the perimeter:

1) Agent $i \in 1 \dots N$ is located at $\lfloor i + \frac{1}{2}(-1)^i \rfloor P/N$
2) Agent $i \in 1 \dots N$ is located at $\lfloor i - \frac{1}{2}(-1)^i \rfloor P/N$

where $\lfloor \cdot \rfloor$ returns the largest integer less than or equal to its argument. The *low-latency exchange configuration* is the behavior realized by the team when oscillating between these two team locations at speed V.

The difference between the two sets of positions in Definition 1 is the sign of the $\frac{1}{2}(-1)^i$ term. The first set of team locations places agent 1 at $x = 0$ and all other agents in pairs at $2P/N$ equal intervals along the perimeter (see Figure 2(c)). The second set of team locations places agent 1 and 2 at position $x = P/N$ and spaces the remaining pairs at $2P/N$ intervals (see Figure 2(b)). Note that for each agent i, the pair of positions in Definition 1 defines the endpoints of the segment on which it remains while in the low-latency exchange configuration.

As indicated earlier, the low-latency exchange configuration is the ideal behavior for a team of agents monitoring a linear perimeter and it will be the desired steady-state behavior of the decentralized algorithm presented in Section IV. In addition to converging to the low-latency exchange configuration, the algorithm will address deletion and insertion of team members and variable length perimeters.

IV. Decentralized Solution

This section presents a decentralized algorithm that causes a team of N agents to reach the low-latency exchange configuration defined in Definition 1. One way to approach the problem is to determine the *coordination variables* [17] or minimum amount of information necessary to achieve cooperation. For this problem, the three critical pieces of information are: (1) the perimeter length, (2) the number of agents on the left side of the perimeter relative to a given agent, and (3) the number of agents on the right side of the perimeter relative to a given agent. When each agent has correct coordination variables, then each will be able to compute the perimeter segment for which it is responsible. The first step in the decentralized solution is to ensure that when each agent has the proper values that coordination will be achieved.

To be precise, let each agent maintain a vector containing its local version of the coordination variables. For each agent $i \in 1 \dots N$, let

$$\xi_i = [P_{R_i} \quad P_{L_i} \quad N_{R_i} \quad N_{L_i}]^T$$

be the coordination vector where P_{R_i} is the length of the perimeter to the right of agent i, P_{L_i} is the length of the perimeter to the left of agent i, and N_{R_i} and N_{L_i} are the number of agents to the right and left of agent i respectively. We adopt the convention that $x = 0$ is the left border of the perimeter and $x = P$ is the right border. An agent i can then calculate the segment for which it is responsible by calculating the perimeter length $P = P_{R_i} + P_{L_i}$, the team size $N = N_{R_i} + N_{L_i} + 1$ and its relative order on the team, $n = N_{L_i} + 1$. By using the definition of the low-latency exchange configuration, the segment for which agent i is responsible is defined by the endpoints at $\lfloor n \pm \frac{1}{2}(-1)^n \rfloor P/N$. We say that each agent has correct coordination variables when for each $i \in 1 \dots N$, $P_{R_i} + P_{L_i}$ matches the true perimeter length, N_{R_i} matches the actual number of agents on the team to the right of agent i, and N_{L_i} matches the actual number of agents on the team to the left of agent i

Consider an algorithm where each agent assumes responsibility for a portion of the perimeter and escorts any of its intruding neighbors to their shared segment border. Algorithm A ensures that if each agent has correct coordination variable values (i.e. each agents knows the length of the perimeter, the total number of agents on the team, and its position in the team), then the low-latency exchange configuration will reached.

if agent i rendezvous with neighbor j **then**
  Calculate team size $N = N_{R_i} + N_{L_i} + 1$.
  Calculate perimeter length $P = P_{R_i} + P_{L_i}$.
  Calculate relative index $n = N_{L_i} + 1$.
  Calculate segment endpoints:
    $\mathcal{S}_i = \left\{ \lfloor n - \frac{1}{2}(-1)^n \rfloor P/N, \lfloor n + \frac{1}{2}(-1)^n \rfloor P/N \right\}$.
  Communicate $\mathcal{S}_i$ to neighbor j and receive $\mathcal{S}_j$.
  Calculate shared border position $p = \mathcal{S}_i \cap \mathcal{S}_j$.
  Travel with neighbor j to shared border position p.
  Set direction to monitor own segment.
else if reached perimeter endpoint **then**
  Reverse direction.
else
  Continue in current direction.
end if

Alg. A. Neighbor Escort — from the Perspective of Agent i

For every consecutive pair of agents, there is a single position where their segments border each other. When each

agent has a knowledge of the length of the perimeter and its order in the team, then the endpoints of the segment for which it is responsible are computed. The endpoint shared with a neighbor is the shared border position to which both will travel together as seen in Algorithm A. In other words, each agent escorts its neighbors to the position at which they should have met had they been in perfect synchronization. Note that agents only reverse direction at perimeter endpoints and when they finish escorting neighbor agents, so each agent is guaranteed to meet its neighbors.

Theorem 1: Let the perimeter length P and number of agents N be fixed. If all agents have correct coordination values, then Algorithm A ensures that the low-latency exchange configuration is achieved after time $2T$ has passed where $T = V/P$ corresponds to the time required for one agent to travel the length of the perimeter.

Proof: Team agents can initially be positioned anywhere along the perimeter and can be traveling either to the left or right (recall that constant uniform velocity is assumed). Since each agent has correct coordination variables, then each can calculate the segment along the perimeter for which it is responsible. Agents are guaranteed to meet both neighbors since Algorithm A only commands agents to reverse direction at a *perimeter* (not segment) endpoint or when concluding a neighbor escort.

For N agents monitoring a border of length P, order the segments of size P/N from the left edge of the perimeter as $1, \ldots, N$ and label each agent so that agent i is responsible for segment i. Consider first the actions of agent 1. Once agent 1 has escorted its right neighbor to their shared border, then no agent to the right of agent 1 will ever travel along segment 1 again. This can be seen by noting that after agents 1 and 2 split at their shared boundary *both* will travel the length of one segment to get to the opposite end of their respective segments. If agent 2 meets agent 3 along the way, then agents 2 and 3 will continue to their shared border before agent 2 reverses direction, as in Figure 3. Therefore, agent 2 will travel at least one segment length away from the boundary between segments 1 and 2. Since both travel at a uniform constant velocity, then agent 1 will arrive back at the border between segments 1 and 2 at the same time or before agent 2, but never after. Now consider agent 2 after it has been escorted by agent 1 to their shared boundary. Since by this time agent 2 never ventures into segment 1, the border between agents 1 and 2 can be regarded as a fixed perimeter endpoint for agent 2. The same analysis now holds if we consider agent 2 as the leftmost agent in a set of $N - 1$ agents. Observe that the same argument holds starting with the rightmost agent and considering all agents to the left. Therefore, there is a time τ after which all agents are constrained to their respective segments. This implies that the low-latency exchange configuration of Section III has been reached.

The worst case situation occurs when all agents are stacked infinitesimally close at one end of the perimeter and are traveling toward the other end. Once T has passed, all agents are at the opposite end of the perimeter where they meet both of their neighbors. Each pair will travel to their shared borders which for the farthest pair will require a travel time less than T. Therefore, the steady-state behavior will be achieved before time $2T$. ■

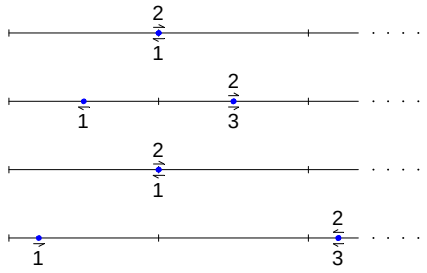

Fig. 3. Possible cases for rendezvous of agent 1 with its neighbor. a) Agents 1 and 2 separate at their shared border. b) Agent 2 encounters agent 3 earlier than expected and escorts it to their shared border before reversing direction. c) Agents 1 and 2 separate at their shared border. d) Agent 2 encounters agent 3 later than expected and escorts it to their shared border before continuing on to meet agent 1.

Figure 4 shows a simple scenarios with 8 agents spreading out over a fixed perimeter where each agent begins with correct coordination variables. In Figure 4 the positions of agents along the perimeter at each instant of time is indicated as follows: for each time instant on the horizontal axis, the agents' placement (from $x = 0$ to $x = P$) are plotted on the vertical axis. The lattice structure indicates that the desired steady-state behavior has been reached since agents turn around precisely at their desired neighbor rendezvous locations. Note that the agents require very few meetings with each other to converge to the proper configuration.

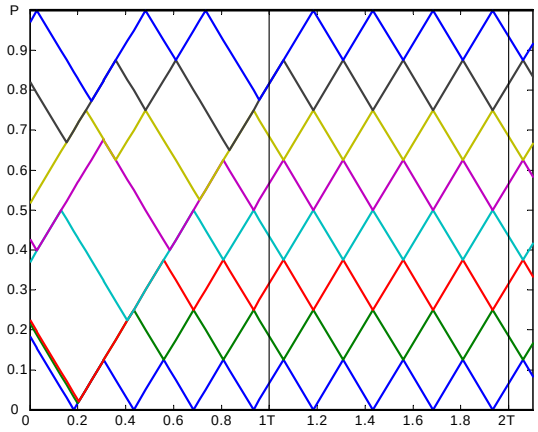

Fig. 4. Example team behavior for point agents whose behavior is governed by Algorithm A. The position of agents along the perimeter is indicated vertically with the time axis shown horizontally. The lattice structure indicates that the desired steady-state behavior has been reached.

Algorithm A is completely decentralized since each agent makes its own decisions rather than being directed by a central team decision maker. A centralized algorithm, using knowledge of the positions of the entire team and the state of the perimeter, can route the team in a way to achieve the low-latency exchange configuration in a shorter time frame. In [18] a centralized perimeter surveillance algorithm is compared to Algorithm A. Monte-Carlo simulations indicate that the centralized algorithm reaches the low-latency exchange configuration on average $0.67T$ seconds faster than the Algorithm A

(standard deviation of $0.17T$ seconds).

There are number of tradeoffs to consider when comparing centralized and decentralized algorithms. Decentralized algorithms are scalable, minimize communication bandwidth, and are robust. Centralized algorithms are efficient (near optimal), can handle coupled and complex tasks, and are usually easier to show provably good performance. The value of Algorithm A is that its performance is comparable to the centralized solution in speed, but is implemented in a scalable, robust way. Additionally, Algorithm A is guaranteed to converge to the optimal steady-state solution with very little communication range. For missions where robustness, scalability, and communication range and bandwidth are valued more than efficiency, Algorithm A provides provably good performance with only a small sacrifice in efficiency when compared to its centralized counterpart.

Summary

This section has detailed a decentralized algorithm and presented a proof that Algorithm A converges to the low-latency exchange configuration in finite-time when each agent has a correct understanding of the size of the perimeter and number of agents on the team. Such an algorithm is scalable, robust, and operates with minimal communication range and bandwidth.

V. Changing Team Size and Perimeter Length

Theorem 1 ensures finite time convergence to the low-latency exchange configuration of Section III when the correct values of the coordination variables are known by each agent. By allowing each agent to update its instantiation of the coordination variables, Algorithm A can be modified to ensure each member of the team will obtain the correct values. This will allow the team to naturally compensate for agent reassignment or loss, and perimeter growth.

Each agent maintains local instantiations of the coordination variables that track the perimeter distance and the number of agents to its left and to its right. These coordination variables are updated when meeting with another agent on the team by querying the neighbor about the portion of the perimeter which it has most recently traveled. If the perimeter and number of agents is fixed, then the coordination variables will eventually be consistent among the team since agents are guaranteed to meet both neighbors. Once the coordination variables are correct, Theorem 1 ensures that the desired steady-state behavior will be achieved. Note that the same method used to update the coordination variables can also be used to detect changes in the perimeter or insertion/deletion of team members.

Algorithm B operates in the same manner as Algorithm A, with the additional steps of communicating and updating the coordination variables. For example, consider two agents starting from opposite ends of the perimeter, each without knowledge of the other. Let agent 1 start at $x = 0$ and agent 2 start at $x = P$, but let the launch time of agent 2 be delayed with respect the launch of agent 1. As each agent

if agent i (left) rendezvous with neighbor j (right) **then**
  Update perimeter length and team size:
    $P_{R_i} = P_{R_j}$
    $N_{R_i} = N_{R_j} + 1$
  Calculate team size $N = N_{R_i} + N_{L_i} + 1$.
  Calculate perimeter length $P = P_{R_i} + P_{L_i}$.
  Calculate relative index $n = N_{L_i} + 1$.
  Calculate segment endpoints:
    $\mathcal{S}_i = \left\{ \lfloor n - \frac{1}{2}(-1)^n \rfloor P/N, \lfloor n + \frac{1}{2}(-1)^n \rfloor P/N \right\}$.
  Communicate $\mathcal{S}_i$ to neighbor j and receive $\mathcal{S}_j$.
  Calculate shared border position $p = \mathcal{S}_i \cap \mathcal{S}_j$.
  Travel with neighbor j to shared border p.
  Set direction to monitor own segment.
else if reached left perimeter endpoint **then**
  Reset perimeter length to the left $P_{L_i} = 0$.
  Reset team size to the left $N_{L_i} = 0$.
  Reverse direction.
else if reached right perimeter endpoint **then**
  Reset perimeter length to the right $P_{R_i} = 0$.
  Reset team size to the right $N_{R_i} = 0$.
  Reverse direction.
else
  Continue in current direction keeping track of traversed perimeter length.
end if

Alg. B. Variable Neighbor Escort — from the Perspective of Agent i

progresses along the perimeter, it keeps track of the distance traveled from launch. When the two agents finally meet, agent 1 updates N_{R_1} to be equal to one plus the number of agents to the right of agent 2 and P_{R_1} equal to P_{R_2} communicated from agent 2; similarly, agent 2 updates N_{L_2} and P_{L_2} using the communication packet from agent 1. At this point, the coordination variables are correct and Theorem 1 ensures that the low-latency exchange configuration will be reached in finite time.

Theorem 2: Let the perimeter length P and number of agents N be fixed. Algorithm B ensures that the low-latency exchange configuration is achieved before $5T$ units of time for arbitrary initial conditions of position, direction, and coordination variables of each agent on the team.

Proof: We first prove that all agents on the team converge to the correct coordination variables in finite time when using Algorithm B. Since an agent only changes direction at perimeter endpoints or when completing a meeting with its neighbors, all agents are guaranteed to meet their neighbors along the perimeter.

Order the N agents from the left edge of the perimeter as $1, \ldots, N$ and consider the actions of agent 1. Agent 1 is guaranteed to visit the left endpoint of the perimeter either after an escort from agent 2 or immediately due to initial conditions. Once agent 1 has visited the perimeter endpoint, both N_{L_1} and P_{L_1} are correct due to the section of Algorithm B that resets those variables at endpoint rendezvous. Now consider the meeting of agent 1 and agent 2. At this point, agent 2 updates N_{L_2} and P_{L_2} through communication with agent 1 and thereby obtains correct values for those coordination

variables. Note that repeated meetings between agent 1 and 2 will not change the correctness of their coordination variables since N and P are fixed. Now consider agent 2 as the left most agent in a team of $N-1$ agents and note that its right neighbor is ensured to obtain correct left coordination variables. Clearly, the same holds from the right end of the perimeter. Since only one neighbor meeting is required after the endmost agent has obtained correct coordination variables and the team size is reduced at each stage and meetings are guaranteed to occur in finite time, the entire team obtains correct coordination variables in finite time.

During the transient period when the team is learning the correct coordination variables, the calculation of the shared segment border is incorrect relative to the low-latency configuration, but consistent among the agents involved in the rendezvous. This can be seen by noting that after both agents have communicated and updated their coordination variables with the other, they each have the same understanding of P and N and can consistently calculate their shared border position. So while they are escorting each other to the (ultimately) wrong position, they are still guaranteed to continue in the correct directions to ensure that each agent meets both its neighbors.

Once the coordination variables are correct for each agent on the team, application of Theorem 1 ensures that the low-latency exchange configuration will be met in finite time.

In evaluating the performance of Algorithm B, we examine the worst case scenario for a bound on the convergence time. The worst case occurs when the overlap of the team is the greatest, that is, when agents are forced to travel together rather than space out along the perimeter.

Consider N agents stacked infinitesimally close at $x = 0$ and headed in the positive direction (see Figure 5a). After time T has passed, agent N will reach the right perimeter endpoint and update P_{R_N} and L_{R_N} to correct values (Figure 5b).

Once agent N has visited the perimeter endpoint, it will rendezvous with the rest of the team (Figure 5c). At this instant, agent 1 has arbitrary values for its coordination variables. Suppose that the coordination variables for agent 1 and agent N are such that the shared border position for agents N and $N-1$ is at $x = \epsilon$ for $\epsilon << P$. In this case, the entire team will again travel the length of the perimeter to $x = \epsilon$ which requires less than another T units of time (Figure 5d).

At this point, agent N separates from the team and begins heading toward the right end of the perimeter. Agents $1 \ldots N-1$ encounter the left endpoint of the perimeter and update to a completely correct set of coordination variables (Figure 5e). With correct coordination variables, the team begins to space out equally along the perimeter. Note that agent $N-1$ will chase agent N the entire length of the perimeter since it will escort agent $N-2$ to their shared border position and then continue to the right.

Agent $N-1$ follows agent N to the right end of the perimeter for T units of time (Figure 5f). Once they meet, agent N will update to correct coordination variables (Figure 5g). At this point, $3T$ has passed and the team satisfies the conditions for Theorem 1 since each has correct coordination variables. By Theorem 1, at most $2T$ additional time is need to converge to the low-latency exchange configuration (Figure 5h). Therefore, $3T$ is an upper bound to convergence of the team to correct coordination variables and $5T$ is an upper bound to convergence of the team to the low-latency exchange configuration.

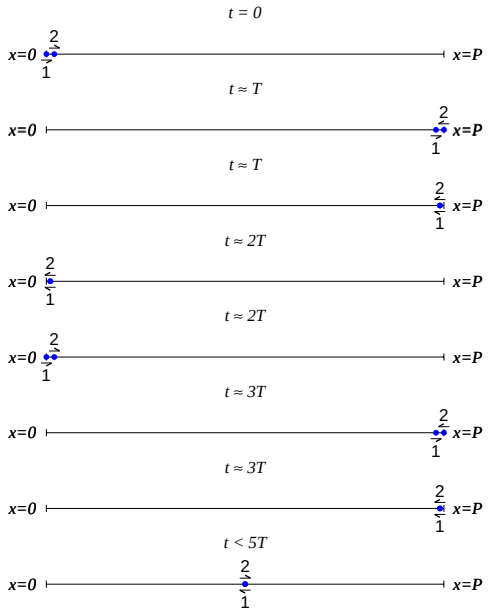

Fig. 5. Worst case scenario for convergence to the low-latency exchange configuration when initial values of the coordination variables are arbitrary. a) Both agents start at one end of the perimeter. b) They travel the full length before agent 2 reaches the right perimeter endpoint. c) Agents rendezvous and calculate shared border position. The worst case is for both to travel together to near the other endpoint. d) The pair reaches the shared border position and separates. e) Agents 1 reaches the perimeter endpoint and updates to correct coordination variables. f) Agent 2 reaches the right perimeter endpoint again while being pursued closely by agent 1. g) The second meeting between agents 1 and 2 where both reach correct coordination variables. h) The team spreads out with a consistent view of the perimeter and number agents on the team.

■

Algorithm B is successful because each agent has finite memory. Since the local instantiations of the coordination variables are updated with the most recent information gathered, past information does not affect team behavior. In addition to enabling the team to converge to correct values of the coordination variables, this finite memory property allows the team to adapt to step changes in perimeter and team size.

Since Algorithm B operates under arbitrary initial conditions, a step change in perimeter or team size would be analyzed by simply considering new initial conditions of the team at the time of the step change. Figure 6 shows agents tracking a changing perimeter and a situation where agents are removed from the team. Algorithm B accommodates step changes in perimeter size, but also allows good tracking for other types of perimeter growth. Note that agents do not have any knowledge *a priori* of the perimeter length or number of agents on the team. The coordination variables of each agent are updated through repeated interactions with other team members.

The ability of Algorithm B to accommodate dynamic changes in team size make the overall system fault tolerant. As agents enter or exit the formation the team adjusts to compen-

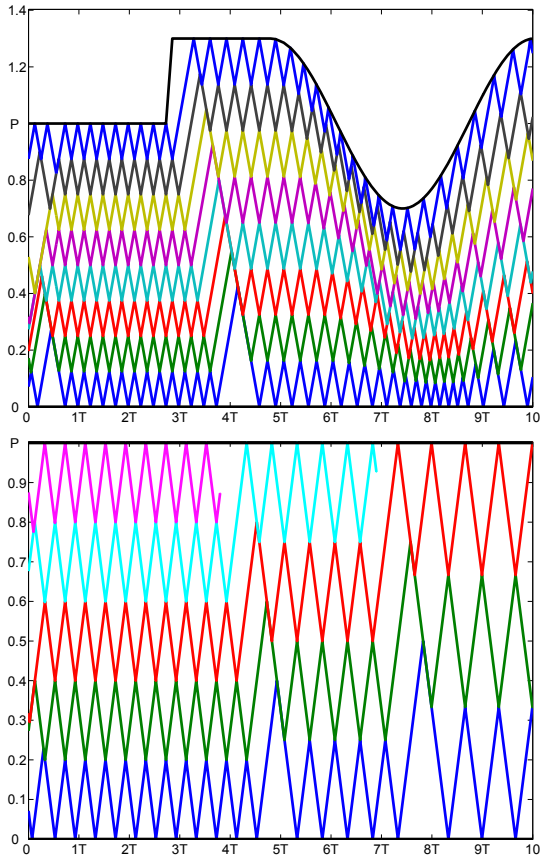

Fig. 6. Team behavior of agents tracking changing perimeters using Algorithm B to continuously update the coordination variables for cases of changing perimeter (top) and changing team size (bottom). Agents learn the size of the perimeter and number of agents on the team through repeated interaction with other team members.

sate naturally. Since each agent only requires a fixed amount of information from its immediate neighbors, Algorithm B has fixed computation and communication requirements regardless of team size. The scalability and fault tolerance of Algorithm B make it an attractive choice for large scale problems.

Summary

This section introduced a simple extension to Algorithm A to allow a team to handle changing perimeter and team sizes. A proof of convergence to the low-latency exchange configuration in the presence of step changes in perimeter and team size was detailed. Due to the ability to accommodate changes in a decentralized way, Algorithm A is noted to be scalable and fault tolerant.

VI. UAV Agents

Algorithm A developed the motion of the reference points for a team of UAVs to follow to achieve the low-latency exchange configuration. In practice, the reference point generalization is only followed when agents are involved in a rendezvous with another agent. Between meetings, the center of the constant airspeed UAV is considered the point along the perimeter. However, since a UAV has a constrained turning radius as noted in Equations (1)–(3), it cannot precisely follow a reference point that can instantaneously change directions. The purpose of this section is to investigate the application of Algorithm A when the curvature constraints of the UAVs are considered.

Consider UAVs flying at constant velocity with nominal turning radius $R = V/\omega_{\max}$ given by the turn rate constraint of Equation (3). A maneuver for reversing direction with constrained turning radius is a simple U-turn. To allow the reference point to follow the pattern dictated by Algorithm A, both UAVs must be able to communicate far enough in advance to begin their U-turn maneuvers so that they complete the maneuver in time to continue following their reference point.

Other methods of rendezvous can be implemented to allow for shorter communication range. For example, the U-turn maneuver could be implemented by having both UAVs circle the point of rendezvous before continuing on in the prescribed direction. This is implemented by having the reference points wait at the rendezvous while the corresponding UAVs loiter at the rendezvous point for a specified amount of time before continuing with Algorithm B and data gathering.

Another option is for neighbors to switch roles at rendezvous. This allows both to continue in their current directions while still maintaining the integrity of the algorithm. When two agents meet, they can negotiate which direction is of higher utility and swap roles if necessary. This would allow UAVs to move down the perimeter toward the base station for refueling without disrupting the perimeter surveillance pattern of the team.

VII. Simulation Results

To verify the feasibility of implementing Algorithm A on a team of UAVs, a high fidelity simulation was performed. Each UAV was simulated with full 6 degree-of-freedom dynamics model with aerodynamic parameters that match the small UAVs flown at BYU [14]. The simulation scenario involved three UAVs monitoring a changing perimeter composed of 4 waypoints with a total length of 1.46 km. Each UAV is equipped with autopilot software that enables accurate waypoint tracking [14] with a turning radius of approximately 50 meters. The communication model allows UAVs to communicate only to adjacent neighbors who are inside the communication range of approximately 370 meters, the minimum distance necessary to perform the U-turn maneuver.

The simulation scenario starts with only two of the three UAVs being launched. Each agent starts without knowledge of the number of agents on the team or the perimeter length. Even though the perimeter is defined by predetermined waypoints, we require the UAVs to initially treat the perimeter length as unknown. After about 400 seconds, a step change in the perimeter length occurred by adding an additional waypoint, followed by another change a short time later. At approximately 900 seconds in simulation time, the third UAV was launched. Before the simulation terminated, the team experienced two more changes in the perimeter length, one at each end.

Figure 7 shows the simulation results by plotting the normalized position of each UAV along the length of the

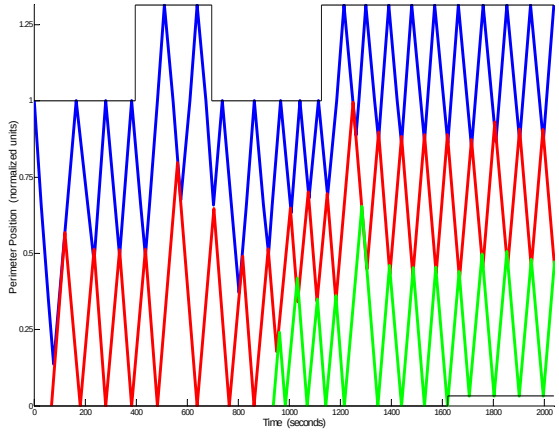

Fig. 7. Simulation results showing the normalized position of each UAV along the perimeter. Changes to the perimeter length occurred at approximately 400, 700, 1100, and 1600 seconds. The third agent was introduced at approximately 950 seconds.

perimeter. Note that in the regions where the team should already be locked into the ideal configuration, some position overlap is still observed. This is caused by the inability of the UAVs to perform the U-turn maneuver precisely, and results in a disturbance to the system. However, the overall behavior of the team is as expected, with the agents reaching the desired steady-state behavior quickly and reacting appropriately to step changes in both the perimeter length and team size.

It should be noted that even though the UAVs cannot turn around instantaneously, the position plot in Figure 7 shows the reference point being followed by the UAV. When the UAV is not implementing a U-turn, the reference point is the center of the UAV; during U-turn maneuvers, the reference point continues along the path to the agreed upon rendezvous point and reverses direction.

VIII. Flight Test Results

The decentralized cooperative-surveillance algorithm, Algorithm B, was further validated by hardware flight tests using the experimental testbed described in Ref. [14]. Figure 8 shows the key elements of our testbed. The first frame shows the Kestrel autopilot which is equipped with a Rabbit 3000 29 MHz processor, rate gyros, accelerometers, and absolute and differential pressure sensors. The second frame in Figure 8 shows the airframes similar to the ones used for the flight tests reported in this paper. The airframe is a 48 inch wingspan Zagi XS EPP foam flying wing that was selected for its durability and adaptability to different mission scenarios. Embedded in the airframe are the autopilot, batteries, a 1000 mW, 900 MHz radio modem, and a GPS receiver. The third frame in Figure 8 shows the ground station components. A laptop runs the Virtual Cockpit software that interfaces through a communication box to the UAVs. An RC transmitter is used as a stand-by fail-safe mechanism to facilitate safe operations.

Flight test results involving two UAVs are shown in Figure 9 which displays the normalized position of two UAVs along the perimeter. Figure 10 shows the inertial position plots that were generated from the actual telemetry files of the UAVs.

Fig. 8. Hardware platform used to obtain experimental results. The first frame shows the Kestrel autopilot designed at BYU. The second frame shows airframes similar to those used in this study. The third frame shows the ground station components for our testbed.

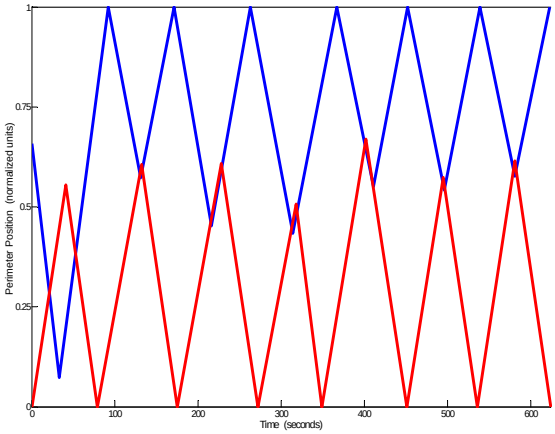

Fig. 9. Experimental results showing the normalized position of each UAV along the perimeter. Algorithm B was started at approximately 50 seconds.

Figure 10 demonstrates the effectiveness of Algorithm B by showing (a) the initial condition for the two agents, (b) the first rendezvous, (c) the turn-around at the shared border, (d) the first meeting of the perimeter endpoints, (e) the second rendezvous, and (f) the second meeting of the perimeter endpoints.

Algorithm B was initiated at approximately 50 seconds, after the two agents had passed each other. The first UAV (blue), having traveled a greater distance than its neighbor, turned around immediately while the second UAV (red) traveled to the shared border before turning around. At this point the agents reached the steady-state configuration. As seen in Figure 9, there is some overlap in position between the two agents. This is a result of the inability of the UAVs to complete a precise U-turn maneuver. It should also be noted that the shared-border position of the two agents appears to be around 60% of the perimeter length instead of the theoretically predicted 50%. This deviation was caused by wind pushing the second agent, thereby enabling Agent 2 to cover more distance than Agent 1. Wind speeds during the flight tests were estimated at 35% of the airspeed of the UAVs. Despite the disturbance of the wind, the agents were still able to effectively distribute themselves evenly along the perimeter.

IX. Conclusions

This paper has presented a decentralized algorithm for perimeter surveillance that converges in finite time. By sharing information regarding the perimeter length and number of team members, each agent obtains a consistent set of coordination variables that allows the decentralized algorithm to

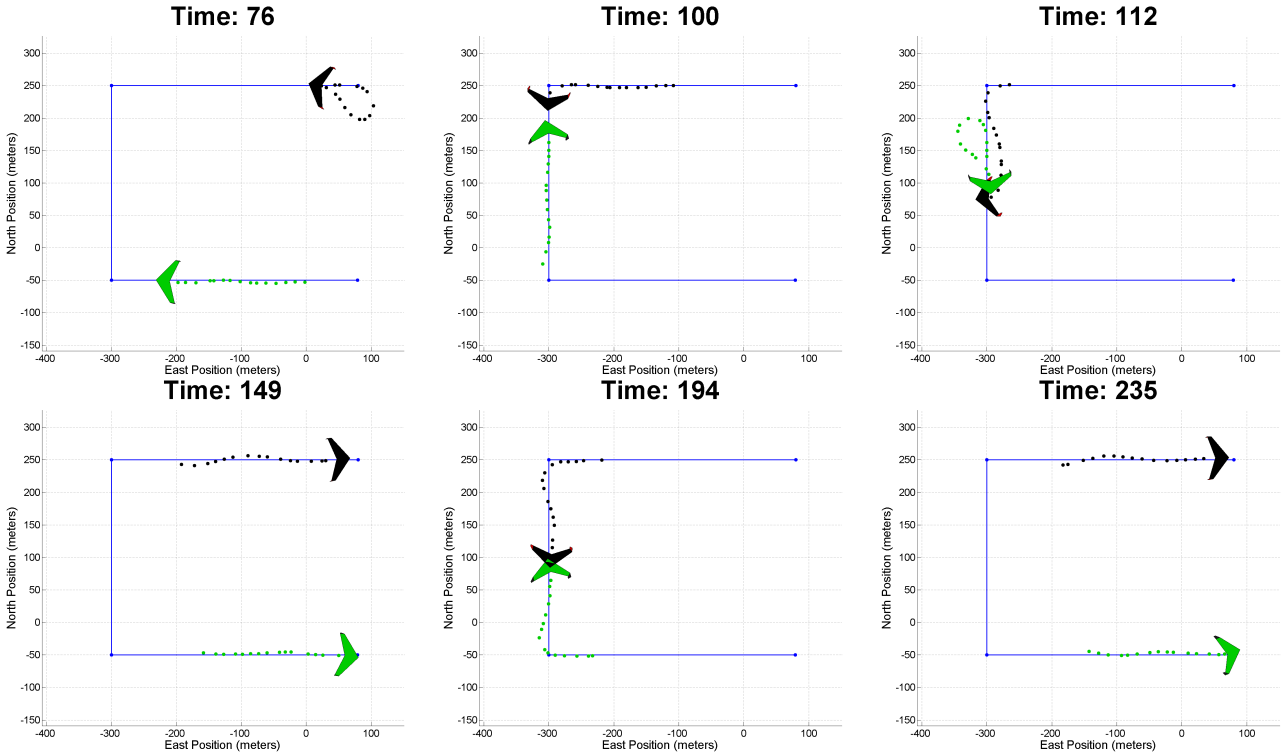

Fig. 10. Various plots generated from the actual telemetry files of the UAVs collected during the experimental flight tests. These demonstrated the functionality of Algorithm B, where (a) are the initial conditions, (b) is the first rendezvous,(c) is the turn-around at the shared border,(d) is the first meeting of the perimeter endpoints,(e) is the second rendezvous, and (f) is the second meeting of the perimeter endpoints.

operate effectively. Advantages of the algorithm include the ability to monitor changing perimeters, account for dynamic insertion and deletion of team members, and the ability to operate with a small communication range in a decentralized manner. The algorithm is limited to a team of homogeneous vehicles on a perimeter homeomorphic to a line and is not as efficient in the transient period as compared to a centralized algorithm. Simulation and flight tests were performed to validate the effectiveness of the algorithm.

REFERENCES

[1] J. Clark and R. Fierro, "Cooperative hybrid control of robotic sensors for perimeter detection and tracking," in *Proceedings of the American Control Conference*, 2005.

[2] B. A. White, A. Tsourdos, I. Ashokoraj, S. Subchan, and R. Zbikowski, "Contaminant cloud boundary monitoring using UAV sensor swarms," *AIAA Journal of Guidance, Control, and Dynamics*, (submitted).

[3] A. L. Bertozzi, M. Kemp, and D. Marthaler, "Determining environmental boundaries: Asynchronous communication and physical scales," in *Proceedings of the Block Island Workshop on Cooperative Control*. Springer-Verlag Series: Lecture Notes in Control and Information Sciences, 2004.

[4] D. W. Casbeer, S.-M. Li, R. W. Beard, T. W. McLain, and R. K. Mehra, "Forest fire monitoring using multiple small uavs," in *Proceedings of the American Control Conference*, 2005.

[5] D. W. Casbeer, D. B. Kingston, R. W. Beard, T. W. McLain, S.-M. Li, and R. Mehra, "Cooperative forest fire surveillance using a team of small unmanned air vehicles," *International Journal of System Sciences*, vol. 36, no. 6, pp. 351–360, May 2006.

[6] R. T. Laird, H. R. Everett, G. A. Gilbreath, T. A. Heath-Pastore, and R. S. Inderieden, "MDARS multiple robot host architecture," in *Association of Unmanned Vehicle Systems, 22nd Annual Technical Symposium and Exhibition*, 1995, available at http://www.nosc.mil/robots/land/mdars/auvsmrha.html.

[7] H. R. Everett, "Robotic security systems," *IEEE Instrumentation & Measurement Magazine*, vol. 6, no. 4, pp. 30–34, Dec. 2003.

[8] Space and Naval Warfare Systems Command, "Mobile detection assessment and response system (MDARS)," http://www.nosc.mil/robots/land/mdars/mdars.html.

[9] M. Kemp, A. L. Bertozzi, and D. Marthaler, "Multi-UUV perimeter surveillance," in *Proceedings of the IEEE/OES Autonomous Underwater Vehicles Conference*, 2004.

[10] C. H. Hsieh, Z. Jin, D. Marthaler, B. Q. Nguyen, D. J. Tung, A. L. Bertozzi, and R. M. Murray, "Experimental validation of an algorithm for cooperative boundary tracking," in *Proceedings of the American Control Conference*, 2005.

[11] S. Susca, S. Martinez, and F. Bullo, "Monitoring environmental boundaries with a robotic sensor network," *IEEE Transactions on Control Systems Technology*, (accepted for publication).

[12] D. B. Kingston, R. S. Holt, R. W. Beard, T. W. McLain, and D. W. Casbeer, "Decentralized perimeter surveillance using a team of UAVs," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2005.

[13] W. Ren, R. W. Beard, and T. W. McLain, *Cooperative Control*. Springer-Verlag Series: Lecture Notes in Control and Information Sciences, 2004, vol. 309, ch. Coordination Variables and Consensus Building in Multiple Vehicle Systems, pp. 171–188.

[14] R. Beard, D. Kingston, M. Quigley, D. Snyder, R. Christiansen, W. Johnson, T. Mclain, and M. Goodrich, "Autonomous vehicle technologies for small fixed wing UAVs," *AIAA Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 1, pp. 92–108, Jan. 2005.

[15] Z. Tang and U. Ozguner, "Motion planning for multitarget surveillance with mobile sensor agents," *IEEE Transactions on Robotics and Automation*, Oct. 2005.

[16] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., 1996.

[17] T. W. McLain and R. W. Beard, "Coordination variables, coordination functions, and cooperative timing missions," in *Proceedings of the American Control Conference*, 2003.

[18] D. B. Kingston, "Decentralized control of multiple UAVs for perimeter and target surveillance," Ph.D. dissertation, Brigham Young University, 2007, http://contentdm.lib.byu.edu/ETD/image/etd2057.pdf.