2007-07-31

Decentralized Control of Multiple UAVs for Perimeter and Target Surveillance

Derek B. Kingston
Brigham Young University - Provo

DECENTRALIZED CONTROL OF MULTIPLE UAVS FOR PERIMETER AND TARGET SURVEILLANCE

by

Derek Bastian Kingston

A dissertation submitted to the faculty of

Brigham Young University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Electrical and Computer Engineering

Brigham Young University

December 2007

BRIGHAM YOUNG UNIVERSITY

GRADUATE COMMITTEE APPROVAL

of a dissertation submitted by

Derek Bastian Kingston

This dissertation has been read by each member of the following graduate committee and by majority vote has been found to be satisfactory.

Date	Randal W. Beard, Chair
Date	Timothy W. McLain
Date	A. Lee Swindlehurst
Date	Michael A. Goodrich
Date	Jeffrey C. Humpherys

BRIGHAM YOUNG UNIVERSITY

As chair of the candidate's graduate committee, I have read the dissertation of Derek Bastian Kingston in its final form and have found that (1) its format, citations, and bibliographical style are consistent and acceptable and fulfill university and department style requirements; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the graduate committee and is ready for submission to the university library.

Date

Randal W. Beard
Chair, Graduate Committee

Accepted for the Department

Michael J. Wirthlin
Graduate Coordinator

Accepted for the College

Alan R. Parkinson
Dean, Ira A. Fulton College of
Engineering and Technology

ABSTRACT

DECENTRALIZED CONTROL OF MULTIPLE UAVS FOR PERIMETER AND TARGET SURVEILLANCE

Derek Bastian Kingston

Department of Electrical and Computer Engineering

Doctor of Philosophy

With the recent development of reliable autonomous technologies for small unmanned air vehicles (UAVs), the algorithms utilizing teams of these vehicles are becoming an increasingly important research area. Unfortunately, there is no unified framework into which all (or even most) cooperative control problems fall. Five factors that affect the development of cooperative control algorithms are objective coupling, communication, completeness, robustness, and efficiency. We classify cooperative control algorithms by these factors and then present three algorithms with application to target and perimeter surveillance and a method for decentralized algorithm design.

The primary contributions of this research are the development and analysis of decentralized algorithms for perimeter and target surveillance. We pose the cooperative perimeter surveillance problem and offer 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. Convergence of the algorithm to the optimal configuration is proven to occur in finite-time. Simulation and hardware results are presented that demonstrate the applicability of the solution.

For single target surveillance, a team of UAVs angularly spaced (i.e. in the splay state configuration) provides the best coverage of the target in a wide variety of circumstances. We propose a decentralized algorithm to achieve the splay state configuration for a team of UAVs tracking a moving target and derive the allowable bounds on target velocity to generate a feasible solution as well as show that, near equilibrium, the overall system is exponentially stable. Monte Carlo simulations indicate that the surveillance algorithm is asymptotically stable for arbitrary initial conditions. We conclude with high fidelity simulation and actual flight tests to show the applicability of the splay state controller to unmanned air systems.

ACKNOWLEDGMENTS

I wish to express sincere appreciation to Professor Randy Beard for his guidance throughout the implementation and writing of this document. In addition, many others in the MAGICC lab have helped make this work possible including James Hall, Sujit, Wei Ren, Ryan Holt, David Johansen, and Greg Aldredge. I also owe a debt of gratitude to my wife, Sarah, for her support and patience.

This research was supported by NASA to Scientific Systems Company, Inc and Brigham Young University under STTR contract number NNA04AA19C, 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.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the recent development of reliable autonomous technologies for small unmanned air vehicles (UAVs), the algorithms utilizing teams of these vehicles are becoming an increasingly important research area. In many cases, a team of small UAVs can accomplish tasks in a more efficient manner than a larger, more capable single vehicle. Of course, methods for cooperative control are used by a variety of autonomous robotic agents with a myriad of potential applications including: chemical spill monitoring [1]; forest fire fighting [2]; search and rescue; exploration (planet, mapping); surveillance [3]; perimeter approximation [4]; manufacturing [5]; maintenance; interferometry [6]; cooperative manipulation/transportation [7]; power systems [8]; sensor networks [9]; cooperative target engagement [10]; radar spoofing [11]; and automated highway systems [12]. It is our intent to investigate the key elements of cooperative control systems and design algorithms that allow teams of UAVs to perform surveillance tasks.

A cooperative control system consists of multiple (often dynamic) agents that share a common objective. In most cases, the objective can only be attained through the sharing of information, tasks, and/or resources. In the broadest sense, a particular cooperative control problem can be categorized by the amount of cooperation needed to fulfill the mission objectives. This "level of cooperation" is determined by a number of factors such as:

1. **Objective coupling**. Many problems *require* multiple agents to satisfy the objective, while other problems can be solved with a single agent, but are more efficiently completed with a team. For example, the Cooperative Moving Target Engagement (CMTE) scenario requires one or more agents to track a target

while a separate agent attacks it [10] – a single agent cannot simultaneously fill both roles and the mission can only be accomplished with a cooperative system. On the other end of the spectrum is the search problem; a single vehicle can search a large area alone, but a team of vehicles can improve the search efficiency. It is the author's assertion that objective coupling is the main factor in determining when a particular cooperative control problem can be solved in a decentralized manner. It also heavily influences the scaling of the algorithm with respect to team size. If many agents are required to have tight coordination or high efficiency requirements, then algorithms will often scale with team size making large teams impractical from a computational complexity stand point.

2. **Level of communication**. A cooperative control problem has at its heart the sharing of information to improve the performance of the team. The communication capabilities and constraints of the team of agents play a large role in determining how much cooperation can be achieved. The two extremes of this element are complete communication where each team member has access to the complete state of all the others; and pure sensory input where each agent acts solely on the data collected locally, such as in a swarm [13]. The level of communication in the system also influences the scalability of the solution. When agents only communicate locally, e.g. to a fixed size neighborhood, adding more agents does not affect the complexity. A cooperative control solution that requires the state of each agent to be known to all other agents requires that bandwidth and computation scale with the size of the team.
3. **Completeness**. It is often important to be able to prove or ensure that a cooperative control algorithm will complete the objective. Many algorithms are based on Monte-Carlo simulations or hardware tests to verify performance, while others can be shown analytically to reach the desired team behavior. In other cases, the cooperative objective is only met in the statistical mean as time evolves. It is important to understand the convergence properties of a particular algorithm when judging its usefulness and applicability.

4. **Robustness**. An algorithm may be provably complete only under a very strict set of operating conditions. In most practical situations, however, it should also be tolerant of disturbances, noise, loss of communication, and agent failure. The ability of an algorithm to deal with these uncertainties often make it superior to a more efficient, but less robust algorithm.
5. **Efficiency**. In many cases, a team of agents is used simply to increase the efficiency at which a problem is completed. Algorithms that utilize optimization techniques to coordinate the team can achieve much greater performance than algorithms that wander and rely on randomness to reach completeness. This element is closely tied to the others; there seems to be a trade-off in many problems between efficiency and robustness. Additionally, algorithms that are highly efficient typically require more communication and are tightly coupled leading to problems of team scaling.

There is no unified framework into which all (or even most) cooperative control problems fall. For this reason, different cooperative control algorithms are difficult to compare in a reasonable manner. Using the above list to categorize algorithms and evaluate them in each area can lead to useful conclusions about the usefulness and applicability of presented algorithms. We place a particular emphasis on the robustness of an algorithm. The natural ability of an algorithm to deal with loss of agents and disrupted communication is very valuable to the practical deployment of an algorithm on a UAV team. The most common method of making a cooperative control algorithm robust is to implement it in a decentralized way. In a decentralized algorithm, agents make decisions based on neighbor interaction without relying on central agent or single point of failure. We recognize, of course, that some algorithms are either extremely inefficient or impossible to implement in a decentralized way, but for problems that allow for a decentralized solution, that solution is often more practical and useful than its centralized counterpart.

In this research, we present three algorithms with application to target surveillance and a method for decentralized algorithm design. In Chapter 2, we present a highly coupled target tracking and prosecution scenario solved by a centralized op-

timization. This algorithm serves as an example of a situation in which completing the mission objectives in an efficient way is difficult without centralization. The algorithm is efficient and complete, but requires full communication and scales exponentially with team size. Additionally, its robustness is poor since the solution must be recomputed when disturbances are encountered. The desire to move to a decentralized algorithm motivates Chapters 3 and 4 in which consensus methods are explored. Chapter 3 proves that most consensus algorithms are input-to-state stable and therefore can be put in cascade with a centralized algorithm to achieve decentralization in some cases. Chapter 4 addresses the average-consensus problem and postulates that when extra bandwidth is available, ad hoc networks may yield better performance.

The main results of this document are contained in Chapters 5 and 6 where we present decentralized algorithms for single target and perimeter surveillance, respectively. Chapter 5 postulates a cooperative splay state controller and proves its completeness. The splay state controller requires only immediate neighbor communication and is completely decentralized and robust. Hardware tests validate its applicability to real UAV systems. Chapter 6 develops a robust perimeter surveillance algorithm that requires very little communication and is optimal in steady-state and near optimal in the transition region. Additionally, the algorithm can be proven to converge in finite-time and shows practicability through flight tests. We note that the decoupled nature of these problems lend the basis for the decentralized algorithms, but their value is apparent nonetheless.

Chapter 2

Assignment Methods for Coupled Multiple Target Tracking

2.1 Introduction

An important element of any autonomous team operation is the ability to assign members of the team to specific roles or tasks. A typical assignment problem involves assigning n tasks to n agents and can be solved efficiently using a linear program [14]. Extensions to this method involve iteratively building up sequences of these assignments to complete a mission [15, 16] or solving a dual maximal network flow problem [14]. These methods benefit from the power of a linear program to quickly solve large problems, but sacrifice the flexibility needed in some assignment scenarios. For example, strict precedence of tasks is difficult to enforce and tasks that require cooperation between agents can be hard to encode. In addition, some agents may have windows of availability in which particular tasks can/cannot be assigned.

In an effort to address some of these concerns, Schumacher et al. [17, 18] developed a Mixed-Integer Linear Program (MILP) that enforces task timing precedence and generates a complete team assignment tour. However, in order to do this, the cost between consecutive tasks must be known *a priori*. Euclidean distance between targets was used as the cost between tasks, but this simplification leads to suboptimal results since the actual cost between tasks can be significantly greater due to the dynamic constraints of the agents.

Assignment problems with strict task precedence have also been addressed by [19, 20] and [21]. In [19], a complete representation of the search space for UAV task assignment is developed which can be searched directly or with a Genetic Algorithm. Turra et al. [20] add the additional complication of moving targets to the strict task precedence assignment problem and present a solution with a pipeline process that

performs the most computationally-intensive tasks off-line, creating an algorithm that is implementable in real time. Finally, Alighanbari et al. [21] present a method to construct tours of sequentiality constrained tasks which can be solved optimally for small problems and with a Tabu search as the computation becomes intractable. Unfortunately, none of these methods provide explicit methods to account for agent availability windows or tasks that are constrained differently than a simple ordering in time. Additionally, highly coupled tasks resist attempts to be decentralized, making methods similar to [22] and [23] difficult to apply.

We present an assignment method that addresses task timing constraints, agent dynamic constraints (implicitly), cooperative tasks, and agent availability time windows in Section 2.2. Extending this method to find a tour of assignments is discussed in Section 2.4 in connection with the application scenario described in Section 2.3. Simulation results are shown in Section 2.5 and Section 2.6 gives conclusions and directions for future work.

2.2 Time-Dependent Assignment Method

In many cooperative scenarios, the ability to assign cooperative tasks (i.e. two or more agents are assigned subtasks at the same target) is critical to mission performance. This assignment is often complicated by the fact that agents may have windows in time when they can complete specific tasks. In a UAV scenario, these windows are typically a product of the underlying dynamic constraints of the vehicle. For example, if a coordinated task requires one agent to track a target while the other attacks it, then one agent may need to adjust its path to wait for its partner. If the path cannot be extended smoothly (e.g. UAV attack paths cannot be extended without discontinuity in some cases [24]) then there are separate intervals in which the task can be done with a period of infeasibility in between. A MILP formulation of the problem allows these constraints to be addressed[1]. Once the problem nomenclature

[1]It has been brought to the author's attention that the problem setup presented here bears striking resemblance to models of air traffic network flow. See, for example, Ref. [25].

has been established, linear constraints are given that satisfy the requirements of time-dependent task assignment.

2.2.1 Nomenclature

Let K be the number of tasks to be performed on each target, V be the number of agents, and N be the number of targets. Also, let $x_{v,k}^{n\,(w)}$ be a binary decision variable indicating that agent v is to perform task k at target n starting in time window (w). Note that the number of decision variables grows as the product of the numbers of agents, targets, tasks, and windows. To allow for the case when there are more agents than tasks to be done, let z_v be a binary decision variable indicating that agent v is to be assigned the null task (assumed to have zero cost). The case when there are more tasks than agents must be addressed in an iterative manner and will be discussed in Section 2.4. To enforce the timing between tasks and to optimize the total time of the assignment, let t_k^n be the time that task k is started on target n.

To represent the cost associated with each of the decision variables, the windows of time when agent v can accomplish task k at target n must be calculated. For each (v, k, n), a set $W_{v,k}^n$ is formed where each element of $W_{v,k}^n$ contains a start time ($T_{v,k}^{n\,\lfloor w}$) of window w and a stop time ($T_{v,k}^{n\,w\rfloor}$). This can be done in *any* manner suitable to the problem definition. It is this flexibility that gives this assignment method its appeal – as long as the underlying path planning can be represented by windows of task availability, this assignment algorithm can be used. Note that a worst-case target prosecution time exists for each target and the maximum of those times is the upper-bound on the final task completion time. Practically, an upper-bound on target completion time will always exist due to fuel constraints. This maximum target completion time, T, allows the formation of timing inequalities that support task precedence.

2.2.2 Constraints

A Mixed-Integer Linear Program is defined by linear constraints and the cost function that describe the problem. Following [17], a set of constraints and associated cost function to accomplish the assignment is presented below.

Non-Timing Constraints

1. Each agent gets exactly one task or goes to the sink

$$\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{w} x_{v,k}^{n\,(w)} + z_v = 1 \qquad \text{for } v = 1 \dots V \quad . \tag{2.1}$$

2. Any target that receives one task, receives all tasks

$$\sum_{v=1}^{V}\sum_{w} x_{v,1}^{n\,(w)} = \sum_{v=1}^{V}\sum_{w} x_{v,k}^{n\,(w)} \qquad \text{for } n = 1 \dots N,\ k = 2 \dots K \quad . \tag{2.2}$$

3. Each target is serviced at most once (in combination with the above constraint, this also ensures that each target receives each task at most once)

$$\sum_{v=1}^{V}\sum_{w} x_{v,1}^{n\,(w)} \leq 1 \qquad \text{for } n = 1 \dots N \quad . \tag{2.3}$$

Timing Constraints

1. Time to start task k at target n must be in window w of agent v if the corresponding decision has been made. Note that these inequalities are trivially satisfied when $x_{v,k}^{n\,(w)}$ is zero, but become tight restrictions on t_k^n when $x_{v,k}^{n\,(w)}$ is

one

$$
\begin{aligned}
t_k^n &\geq T_{v,k}^{n\,\lfloor w} - (1 - x_{v,k}^{n\,(w)})2NT \\
t_k^n &\leq T_{v,k}^{n\,w\rfloor} + (1 - x_{v,k}^{n\,(w)})2NT \\
\text{for } k = 1 \ldots K,\, v = 1 \ldots V,\, n = 1 \ldots N,\, w \subset W_{v,k}^n \quad .
\end{aligned} \tag{2.4}
$$

2. If precedence of tasks is required, then constraints similar to the following will be needed. Here we constrain the tasks to occur in order $1 \ldots K$

$$
\begin{aligned}
t_k^n &\leq t_{k+1}^n \\
\text{for } n = 1 \ldots N,\, k = 1 \ldots K{-}1 \quad .
\end{aligned} \tag{2.5}
$$

   To have agents cooperate in servicing a target simply requires defining the relative timing of tasks. If, for example, two UAVs were to start two cooperative tasks (say task 1 and 2) simultaneously, then the constraint $t_1^n = t_2^n$ could be added. Similarly, if a task must occur within some interval after a previous task is performed, a constraint pair like $(t_2^n \leq t_1^n + \alpha,\, t_2^n \geq t_1^n)$ is applied.

3. To ensure that as many targets as possible are serviced, we impose a missed target penalty. Note that all t_k^n that are associated with targets that are missed are equal to MT where M is the number of missed targets. This constraint also eliminates the degenerate solution of assigning all agents to the null task

$$
\begin{aligned}
t_k^n \geq & \left(\sum_{m=1}^{N} \left\{ 1 - \sum_{v=1}^{V} \sum_{w} x_{v,k}^{m\,(w)} \right\} \right) T \\
& - \left(\sum_{v=1}^{V} \sum_{w} x_{v,k}^{n\,(w)} \right) 2NT \\
t_k^n \leq & \left(\sum_{m=1}^{N} \left\{ 1 - \sum_{v=1}^{V} \sum_{w} x_{v,k}^{m\,(w)} \right\} \right) T \\
& + \left(\sum_{v=1}^{V} \sum_{w} x_{v,k}^{n\,(w)} \right) 2NT \\
& \text{for } k = 1 \ldots K,\, n = 1 \ldots N \quad .
\end{aligned} \tag{2.6}
$$

2.2.3 Cost Function

Reasonable cost functions for many assignment scenarios include minimizing the final task completion time for all targets

$$J = \sum_{n=1}^{N} t_K^n \tag{2.7}$$

or minimizing all task completion times for all targets

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} t_k^n \ . \tag{2.8}$$

2.3 UAV Assignment Scenario

One scenario which requires a high level of cooperation between team members and has the additional complexity of nonlinear agent dynamics is the Cooperative Moving Target Engagement (CMTE) scenario. CMTE requires that two or more UAVs track a moving (ground) target with doppler radar while an additional UAV launches a GPS-guided munition. The sensed target position and associated error ellipse from each tracking UAV are fused to form a precise GPS location of the target for the munition to follow. To reduce the error in the location of the moving target, the UAVs tasked to perform the tracking must have different line-of-sight angles to the target, preferably near orthogonal views. In addition, a moving target can only be detected and tracked if the UAV has a line-of-sight view to the target within some offset angle, γ, from the heading of the moving target, ψ. Figure 2.1 shows the heading of the target and the associated regions in which UAVs can be located to detect its motion.

Complicating matters further, each UAV has a sensor footprint in which targets must be located to be tracked. The footprint has minimum and maximum ranges and bearings and, due to the configuration of the radar antenna array, is pointed out the wing of the UAV. Figure 2.2 shows a UAV tracking a target and the associated

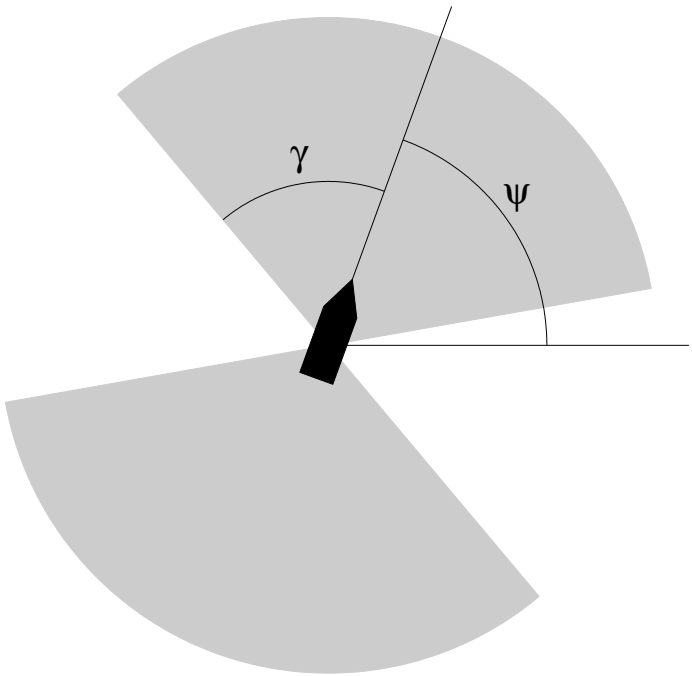


Figure 2.1: Region of detectability based on target heading.

sensor footprint relative to the orientation of the UAV. The sensor can scan on either side of the UAV, but not both at the same time.

A team of UAVs designated to track and prosecute an area of targets can also be supported by an additional "off-board" team member who is located outside of the area and has a powerful sensor with an assumed 360-degree sensor footprint able to view the entire field. This assumption can be relaxed, but requires additional timing constraints. For purposes of this assignment algorithm, the off-board vehicle is assumed to travel circularly so that its line-of-sight to targets is approximately fixed. UAVs inside the area can then cooperatively track a target with the off-board vehicle or with an inside team member. Because the error in the position of the moving target can be reduced by multiple separated line-of-sight angles to the target, we restrict the difference in bearing angles of the UAVs to the target to be greater than 45 degrees. This restriction partitions the detectability region of the target further into regions that satisfy both the target detectability requirement and the angle offset requirement. For fixed target heading and position and fixed off-board

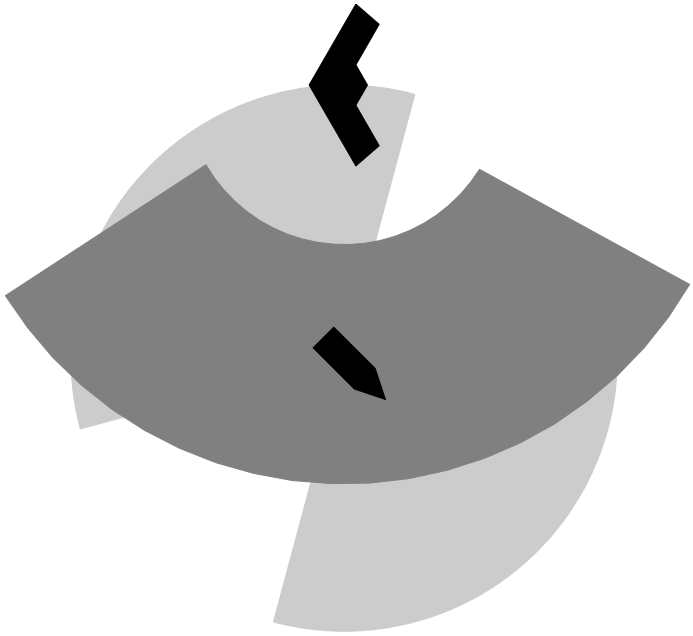


Figure 2.2: UAV sensor footprint (dark gray region).

vehicle position, regions where a target can be cooperatively tracked can be identified and used to develop path-planning routines of the UAVs to complete the mission.

While the bulk of the complexity in the CMTE scenario comes from the cooperative tracking of targets, the attack on the target must also be considered. All UAVs inside the area of interest can track targets and drop weapons. To be in position to attack, a UAV must be headed toward the target and be within a maximum and minimum range. Once the weapon is launched, the attacking UAV is free to be reassigned to other tasks, but the UAVs tracking the target must track the target for the duration of the weapon flight.

The CMTE scenario requires strict timing constraints between tasks and cooperation between team members. UAV dynamics impose constraints on path planning techniques to get into tracking and attacking positions. This complexity makes the CMTE scenario an ideal problem for studying time-dependent cooperative assignment methods.

2.4 CMTE Application

To reduce the CMTE scenario to a more reasonable level, the following assumptions and restrictions will be added.

1. Targets have constant heading. Admittedly, this is a poor assumption, but for targets traveling along known roads, it may be justified. Allowing dramatic target heading changes requires many more UAVs to be assigned to track the target to ensure coverage over all possible headings.

2. Tracking of targets occurs along arcs of a circle centered at the target with radius so as to place the target in the center of the sensor footprint (see Fig. 2.2). This allows the path planning to be performed as if the target were stationary since the sensor footprint is much larger than the distance traveled by the target during a typical scenario.

3. Weapons are launched at a fixed distance from the target and an upper bound on the flight time of the weapon is known so as to fix the amount of time after an attack has occurred that the target must be tracked. This simply translates to planning more time to track than will actually be needed for the weapon launch and travel.

These restrictions and assumptions simplify the level of path planning needed to accomplish a CMTE mission. Additional complexity could be added without changing the method of assignment as long as the interface between the nonlinear path planning and the assignment algorithm remains abstracted to the specification of windows of availability of team agents.

Because the CMTE scenario hinges on the ability of UAVs to cooperatively track a moving target, much of the assignment complexity is involved with determining which team members are assigned to track which target and with whom. To this end, the basic time-dependent assignment algorithm developed in Section 2.2 is augmented with additional decision variables to allow pairwise decisions. Let $y_{u,v}^{n\,(w)}$ be a binary decision variable indicating that UAVs u and v are assigned to cooperatively

track target n in time window w. This allows the path planning routines to calculate windows of time when the pair of vehicles (u, v) can cooperatively track a target.

Following the nomenclature established above and in Section 2.2, let $k = 1$ be designated the *attack* task and $k = 2$ be the *track* task, then the following constraints are used to assign a team of UAVs in the CMTE scenario.

2.4.1 Non-Timing Constraints

1. Each agent gets exactly one task or goes to the sink. An agent can be assigned to cooperatively track a target with the off-board vehicle ($x_{v,2}^{n\,(w)}$) or with another inside team member ($y_{u,v}^{n\,(w)}$), but not both. At a higher level, an agent could be assigned to attack ($x_{v,1}^{n\,(w)}$) or track, but not both

$$\sum_{n=1}^{N}\sum_{k=1}^{K}\sum_{w} x_{v,k}^{n\,(w)} + \sum_{n=1}^{N}\sum_{u=1,u\neq v}^{V}\sum_{w} y_{u,v}^{n\,(w)} + z_v = 1 \qquad (2.9)$$
$$\text{for } v = 1 \ldots V \quad .$$

2. Any target that receives one task, receives all tasks. Since the *track* task occurs in two different decision variables, the sum of both must equal the decision to complete the *attack* task by another vehicle

$$\sum_{v=1}^{V}\sum_{w} x_{v,1}^{n\,(w)} = \sum_{v=1}^{V}\sum_{w} x_{v,2}^{n\,(w)} + \sum_{u=1}^{V-1}\sum_{v=u+1}^{V}\sum_{w} y_{u,v}^{n\,(w)} \qquad (2.10)$$
$$\text{for } n = 1 \ldots N \quad .$$

3. Each target is serviced at most once (in combination with the above constraint, this also ensures that each target receives each task at most once)

$$\sum_{v=1}^{V}\sum_{w} x_{v,1}^{n\,(w)} \leq 1 \qquad (2.11)$$
$$\text{for } n = 1 \ldots N \quad .$$

2.4.2 Timing Constraints

The CMTE scenario requires that a target be continuously tracked for the duration of the weapon flight after the weapon has been launched. Since it is assumed that the weapon is launched from a specific distance from the target and the speed of the munition is known, the path planning routines can return windows of time when an agent (or pair of agents) can start tracking a target and continue for at least t_α where t_α is the time from munition launch to impact. This also allows the compaction of t_1^n and t_2^n to t^n since we can constrain the tracking to begin when the weapon is launched without changing the requirements of the scenario.

1. Time to prosecute target n must be in window w of agent v if the corresponding decision has been made

$$\begin{aligned} t^n &\geq T_{v,k}^{n\,\lfloor w} - (1 - x_{v,k}^{n\,(w)})2NT \\ t^n &\leq T_{v,k}^{n\,w\rfloor} + (1 - x_{v,k}^{n\,(w)})2NT \end{aligned} \tag{2.12}$$

$$\text{for } k = 1 \ldots K,\ v = 1 \ldots V,\ n = 1 \ldots N,\ w \subset W_{v,k}^n$$

$$\text{and}$$

$$\begin{aligned} t^n &\geq T_{u,v}^{n\,\lfloor w} - (1 - y_{u,v}^{n\,(w)})2NT \\ t^n &\leq T_{u,v}^{n\,w\rfloor} + (1 - y_{u,v}^{n\,(w)})2NT \end{aligned} \tag{2.13}$$

$$\text{for } u = 1 \ldots V{-}1,\ v = u{+}1 \ldots V,$$

$$n = 1 \ldots N,\ w \subset W_{u,v}^n \quad .$$

2. Due to the reduction of timing variables to the representative target prosecution time, no additional task timing constraints are needed.

3. Missed target penalty carries over from Section 2.2 to ensure that all UAVs are not assigned to the null task. Note that only $x_{v,1}^{n\,(w)}$ needs to be examined due to the coupling through constraint (2.10), which ensures that targets that are not attacked will not be tracked either.

2.4.3 Cost Function

The CMTE scenario simply requires that all targets are prosecuted as quickly as possible, so

$$J = \sum_{n=1}^{N} t^n \ . \tag{2.14}$$

The constraints given in Sections 2.4.1 and 2.4.2 in connection with the cost function in 2.4.3 define a Mixed-Integer Linear Program suitable for assigning each UAV in a team to one task in a CMTE scenario. The completion of the mission is when all the tasks are completed, so the assignment algorithm must be iterated to produce a tour of assignments. A completely optimal solution would require the optimization of path planning and assignment to be coupled. By applying this assignment scheme, a tour of assignments will be suboptimal, but the freedom allowed by separating the path planning and the assignment is desirable. By augmenting the assignment method with an additional set of complete target ordering constraints, heuristics can be used to improve the iterative solution. Specifically, we use the solution to a Traveling Salesman Problem with the targets as cities to guide the iteration, since the distance between targets is a good indication of the spatial coupling of the scene.

The time windows calculated in the underlying path planning routines can easily be shifted by the amount of time a vehicle has already committed to, so an iteration can be used where the state (position and time committed) of the vehicles is updated after each assignment stage is run. Once the assignment algorithm has allocated a task to each vehicle in the team, the earliest target prosecution time is selected. The target corresponding to that time is removed from the target list and the vehicles assigned to the tasks pertaining to the prosecution of that target have their states updated to reflect the time it will take to complete their respective tasks. Vehicles associated with tasks related to the prosecution of other targets are not updated. New time windows are computed with the updated positions and shifted by the amount of time committed to earlier stages. The assignment algorithm is iterated until all targets have been prosecuted.

Obtaining a solution to any MILP formulation is *NP*-hard. This assignment algorithm is based on MILP and, so, is also *NP*-hard, resulting in extreme amounts of computation needed for large problems. A number of different random CMTE situations were simulated to estimate the computation required for various problem sizes. It was found that problems in which the sum of the number of vehicles and targets is less than or equal to 12 are solvable in less than a minute on a modern desktop computer (we used MATLAB and an open-source linear programming package, GLPK). For many CMTE missions, small problem sizes are typical, involving 5 UAVs and 3 to 4 targets. Larger problems will require a re-formulation of the assignment algorithm or a partitioning of the team and target space into problems small enough to be solved in a timely manner. Solving a CMTE mission in small stages has the additional advantage that many of the assumptions used to simplify the path planning (e.g. constant heading) will be invalidated for lengthy tours of target prosecution, and hence, only a small number of targets will be prosecuted (and assigned) at each stage.

2.5 Simulation Example

To illustrate the capability of the time-dependent cooperative assignment algorithm presented in Section 2.4, a problem of size $V = 5$ and $N = 3$ was simulated. UAVs and targets were randomly distributed over an area 110 km wide and 170 km long with an additional off-board vehicle fixed directly north of the area of interest. Figure 2.3(a) shows the initial positions of the targets and in-area UAVs. Each target is shown with an associated detectability region (outlined in black) and cooperative tracking region (solid wedge). Recall that the cooperative tracking region is the intersection of the detectability region with the line-of-sight angles greater than 45 degrees different from the off-board vehicle line-of-sight. Task time availability windows are computed based on computing minimum time trajectories to these regions.

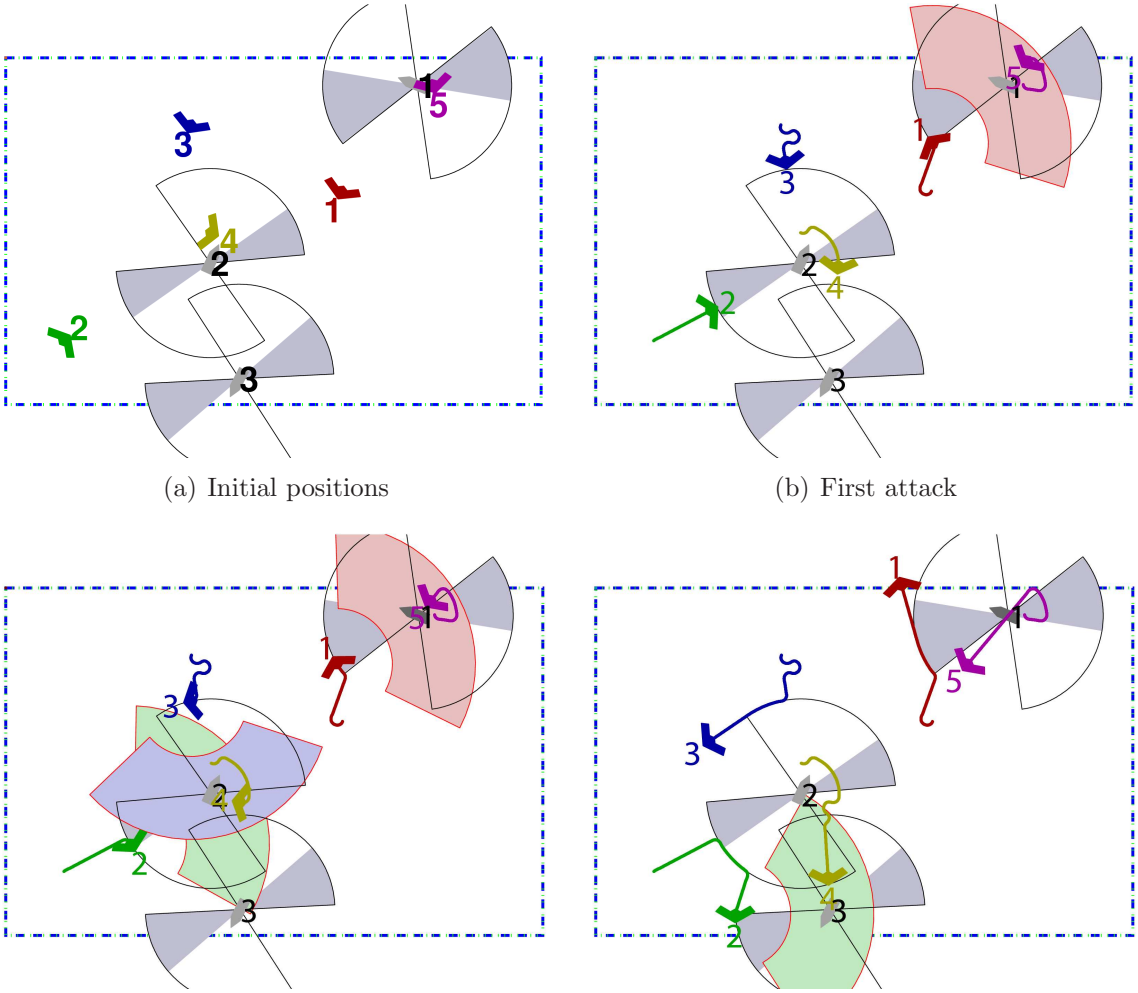
The CMTE scenario is rich in timing complexity making visualization difficult without animation. Figures 2.3(a)-2.3(d) show the progression of the simulated scenario at 4 distinct points in time. Figure 2.3(b) shows that UAV 1 is assigned to

track target 1 in cooperation with the off-board vehicle while UAV 5 attacks. The sensor footprint of the tracking UAV is shown to validate that the UAV is in position to track the target. Because UAV 1 is in the cooperative tracking region of target 1, no other UAVs are needed to track this target. This represents one iteration of the assignment algorithm. During the next iteration, UAVs 2 and 3 are assigned to cooperatively track target 2. Figure 2.3(c) shows the instant in time when UAV 4 releases a weapon to attack target 2. Note that the assignment algorithm correctly assigned 2 UAVs to track this target due to the distance needed for UAV 2 or 3 to reach a cooperative tracking region with sufficient room to track the weapon for the entire weapon flight. Also note that UAV 3 extended its path to arrive at the correct position and time to track the target. Since the algorithm ensures that the target prosecution time falls in the availability time windows of each UAV, no vehicle will be given requirements that violate underlying dynamic constraints. The final target is attacked by UAV 4 with UAV 2 assigned to track in cooperation with the off-board vehicle (Fig. 2.3(d)).

This CMTE mission needed 70 variables (67 binary, 3 continuous) and 81 constraints to describe the optimization and required slightly less than 0.2 seconds to solve the MILP formulation. Current path planning routines are scripted in MATLAB and, for this scenario, required about 10 seconds of computation. It is anticipated that the path planning routines will require significantly less computation as code is moved from MATLAB to C; additionally, for large problems, the optimization will be the most time-intensive part.

2.6 Conclusions and Future Work

An assignment algorithm capable of dealing with agent availability time intervals and explicit task precedence was presented. The flexibility allowed by abstracting the agent path planning from the assignment algorithm allows for complex assignment scenarios to be considered. The Cooperative Moving Target Engagement (CMTE) scenario was presented as an example of a situation in which traditional assignment algorithms are not sufficient. An assignment formulation for the CMTE scenario was



(a) Initial positions (b) First attack

(c) Second attack (d) Final attack

Figure 2.3: Simulated CMTE scenario.

presented, along with a discussion of issues related to task tours and computation requirements.

Future work in this area involves finding heuristics to guide the selection of target ordering as well as partitioning strategies to break large problems into computationally tractable ones. Tighter integration of the linear elements of path planning and the assignment algorithm are also being investigated in an attempt to gain back the optimality lost in the problem assumptions.

Chapter 3

Consensus: Input-to-State Stability

3.1 Introduction

Replacing large, expensive, monolithic vehicles with teams of networked vehicles, promises less expensive, more capable systems. In addition, there are applications where a team of vehicles can accomplish objectives that would be impossible for a single vehicle [26, 27, 10]. To a large extent, the ability of team members to coordinate hinges on their agreement upon a set of information that we call the *coordination variable* [28]. When this information is the same between team members, centralized coordination algorithms (replicated on each agent) can be used to achieve cooperation in a decentralized manner. Unfortunately, in most real-world applications, perfect synchronization is not possible necessitating algorithms that ensure that team members eventually come to a consensus on the value of the coordination variable. Many good approaches solve the consensus problem with varying levels of assumed agent connectivity.

Distributed algorithms for reaching consensus have been a topic of study for many years [29]. Many authors focus on networks of agents that are fully connected. Since the basic consensus problem is trivial in that setting by simply having every agent communicate their data with all others and vote on the outcome value, researchers have focused on dealing with rogue agents in the network that either act arbitrarily or maliciously (e.g. Byzantine Generals Problem [30]). One algorithm that addresses distributed binary decision making (e.g. database commit) in environments with likely changing connectivity is Paxos [31]. When a majority of agents collectively make a decision, Paxos ensures that eventually all other agents will agree with that decision when their connectivity is restored. While the connectively requirements are

certainly more relaxed than assuming full connectivity of the group, the problem of ensuring that a majority of the agents are fully connected to make the initial decision is still in question. It is this aspect of limited connectivity (i.e. agents can only talk to their immediate neighbors and frequently leave the network) that recent literature has begun to address.

Our primary interest in consensus is the construction of an algorithm that accounts for frequent loss of agent connectivity and low bandwidth communication. Many researchers restrict the connectivity of the team by assuming that agents only send information to their immediate neighbors, never explicitly forwarding packages through the network. Each agent updates its value with consistent rules to ensure that by repeatedly receiving communication from neighbors, the entire group will converge to a single value. One of the initial papers along these lines describes how birds and fish could achieve flocking behavior (i.e. the group as a whole travels in the same direction) through nearest-neighbor interaction [32]. In this work, agents updated their heading to be the average of their neighbors' headings where a fixed communication topology was assumed. Numerous other researchers have investigated the role of the group communication topology in reaching consensus using a linear update mechanism in both continuous-time [33, 34] and discrete-time [35]. The primary result from this research states that a group of agents can reach consensus if and only if the union of the graph representing the connectivity of the agents achieves a spanning tree frequently. These ideas were applied to leaderless vehicle formation control [36, 37, 38] and distributed filtering [39, 40]. Notable publications extending consensus protocols to account for delay and/or switching topologies are [41, 42, 43, 44, 45, 46]. In [47], a consensus scheme motivated by the Kalman filter is presented and shown to guarantee asymptotic consensus and explicitly account for relative agent reliability. In this chapter we show that the Kalman consensus scheme and most other from recent literature are input-to-state stable (ISS) with respect to communication noise, and use this fact to design cooperative timing strategies for unmanned air vehicles (UAVs).

UAV cooperative timing problems have been investigated recently in the context of battlefield scenarios where the UAVs are required to converge to the boundary

of a radar detection area to maximize the element of surprise [48, 49, 50, 51, 52]. Cooperative timing problems also arise in refueling scenarios, fire and hazardous material monitoring, moving area of regard problems, and continuous surveillance problems. In this chapter we will investigate a simplified cooperative timing problem that must be accomplished in the presence of an unreliable, dynamically changing communication topology.

In the case of cooperative timing problems, the coordination information is the time-over-target for the whole team. We are particularly interested in the relationship between the consensus algorithm and the cooperative control scheme. Specifically, if the action of each UAV is based on the dynamically changing, local instantiation of the perceived time-over-target, will the team cooperation objective still be achieved?

The main contribution of this chapter is to derive sufficient conditions for the coordination scheme when it is used in connection with an asymptotically stable consensus algorithm. Specifically, we wish to investigate overall system behavior when a cooperative control scheme, designed to be stable when the coordination variable is known *a priori*, is instead, given an estimate of the coordination variable by a consensus scheme. The application of these ideas will be investigated in the context of cooperative timing scenarios.

This chapter is organized as follows. An overview of the Kalman consensus scheme is given in Section 3.2. The Kalman consensus scheme is shown to be input-to-state stable (ISS) in Section 3.3 and this is used to derive a design principle for distributed cooperation algorithms. These principles are applied to a cooperative timing example in Section 3.4.

3.2 Kalman Consensus

In a companion paper, we present a Kalman-filter-inspired technique for consensus seeking [47]. The purpose of the Kalman consensus scheme is to explicitly account for relative agent reliability while at the same time obtaining consensus in the presence of a dynamically changing communication topology. Some of the main results are presented here to facilitate the analysis later of the stability properties of

Kalman consensus. As a matter of notation, we are considering *asymptotic consensus* in the sense that consensus is said to be achieved asymptotically if $\|\xi_i(t) - \xi_j(t)\| \to 0$ as $t \to \infty$ for each pair of agents (i, j), where ξ_i is the i^{th} agent's estimate of the coordination variable whose value all agents must agree upon.

The following update equations describe the Kalman consensus scheme for the i^{th} agent:

$$\dot{P}_i = -P_i \left[\sum_j g_{ij}(t)(P_j + \Omega_{ij})^{-1} \right] P_i + Q_i \tag{3.1}$$

$$K_{ij} = P_i(P_j + \Omega_{ij})^{-1} \tag{3.2}$$

$$\dot{\xi}_i = \sum_{j=1}^{n} g_{ij}(t) K_{ij} \left((\xi_j + \nu_{ij}) - \xi_i \right) \tag{3.3}$$

where ξ_i is i^{th} agent's coordination variable and P_i the associated relative uncertainty for ξ_i. $g_{ij}(t)$ captures the connectivity between agent i and j, specifically, when $g_{ij}(t) = 1$, agent i receives communication from agent j, otherwise $g_{ij}(t) = 0$. ν_{ij} is the noise on the communication channel from agent j to i (assumed to be zero-mean Gaussian with covariance Ω_{ij}). Finally, Q_i is the covariance associated with the zero-mean Gaussian random variable which corrupts the state-space model in a typical Kalman filter setting.

Theorem 1. *Under switching interaction topologies, the Kalman consensus scheme given in Equations* (3.1)–(3.3) *achieves asymptotic consensus if there exist infinitely many consecutive uniformly bounded time intervals such that the union of the interaction graph across each interval has a spanning tree.*

Theorem 1 is proven in [47], but deserves mention here to highlight the conditions under which the Kalman consensus scheme achieves agreement between agents. The central condition (from [53]) is that under dynamically switching communication topologies, a spanning tree of the communication topology graph must be reached infinitely many times. A spanning tree is the least restrictive graph arrangement that includes all agents in a way that allows for consensus. Each time a spanning

tree is achieved, the consensus error is driven closer to zero, so if a spanning tree is reached infinitely many times, then each agent's estimate of the coordination variable approaches the others' asymptotically. The proof of Theorem 1 also shows that the transition matrix in each interval in which a spanning tree is reached is indecomposable and aperiodic, meaning $\lim_{n\to\infty} P^n = \mathbf{1}y^T$, where y is a column vector [54]. Such matrices are part of the stochastic, irreducible, and aperiodic (SIA) class of matrices. SIA matrices are composed of all non-negative entries, have a row sum of 1 and are essentially *averaging* matrices in the sense that a vector operated on by an SIA matrix returns a new vector whose elements are composed of a weighted average of all the entries of the original vector. It is this fact that allows us to conclude uniformity in Section 3.3.

3.3 Consensus Algorithms are Input-to-state Stable

We are primarily interested in the application of consensus algorithms to cooperative control problems. In this chapter we will explore a control architecture where a consensus algorithm is in cascade with a coordination algorithm, as shown in Figure 3.1. Our purpose in this section is to derive conditions on the consensus and coordination algorithms that guarantee that the cooperation objective is achieved. Toward that end, rewrite Equation (3.3) as

$$\dot{\xi}_i = \sum_{j=1}^{n} g_{ij}(t)K_{ij}\left(\xi_j - \xi_i\right) + \sum_{j=1}^{n} g_{ij}(t)K_{ij}\nu_{ij}. \tag{3.4}$$

Defining the total consensus error vector x as $x_{ij} = \xi_i - \xi_j$ and

$$x = (x_{11}, x_{12}, \ldots, x_{1n}, x_{21}, \ldots, x_{nn})^T,$$

we get the state-space model

$$\dot{x} = A(t)x + B(t)\nu \tag{3.5}$$

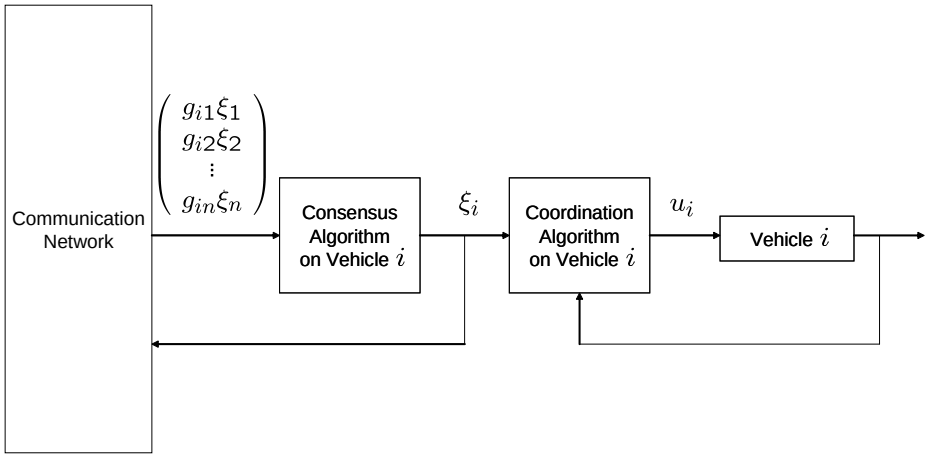


Figure 3.1: The control architecture consists of a consensus algorithm in cascade with a coordination algorithm. The consensus algorithm receives information from the communication network to produce a value of the coordination variable ξ_i. The coordination algorithm uses the coordination variable ξ_i to produce a command the the vehicle u_i. We assume that the same consensus and coordination algorithms are implemented on each vehicle.

where ν is a column vector created by stacking the communication noise terms ν_{ij}, and the elements of $A(t)$ and $B(t)$ are linear combinations of $g_{ij}K_{ij}(t)$ and can be easily constructed from Equation (3.4).

We now state the main technical result of the chapter.

Theorem 2. *Under the hypothesis of Theorem 1, the Kalman consensus scheme given by Equations* (3.1), (3.2), *and* (3.5) *is input-to-state stable.*

The proof of this theorem requires the following two lemmas.

Lemma 3. *Under the hypothesis of Theorem 1, if the communication error ν is zero, then the consensus error x is uniformly stable.*

Proof: As shown in [47], the transition matrix associated with the coordination variable dynamics is SIA. When $g_{ij}(t) = 1$, the i^{th} coordination variable is updated to a weighted average of all agents' coordination variables communicating with i. Since a weighted average can never be greater (or smaller) than any one of the components in the average, the updated ξ_i must be within $[\min(\xi_j), \max(\xi_j)]$. Since all agents that receive communication with other agents use the same averaging scheme, $\xi_i(t) \in$

$[\min(\xi_j(t_0)), \max(\xi_j(t_0))]$ for all t and i. Then

$$\|x\|_\infty \leq \|x(t_0)\|_\infty, \quad \text{for} \quad t \geq t_0.$$

■

Lemma 4. *The norm of $B(t)$ in Equation* (3.5) *is bounded.*

Proof: Since $B(t)$ is composed of linear combinations of $K_{ij}(t)$, if $\|K_{ij}(t)\|$ is bounded for each (i,j), then $\|B(t)\|$ will also be bounded. Referring to Equation (3.2) and recalling that $\Omega_{ij} > 0$ and $P_j(t) > 0$, then $\|K_{ij}\|$ will be bounded if $\|P_i(t)\|$ is bounded. Using Equation (3.1) and noting that $P_i > 0$, Q_i is bounded and $-P_i \left[\sum_j g_{ij}(t)(P_j + \Omega_{ij})^{-1}\right] P_i \leq -P_i(P_i + \Omega_{ii})^{-1} P_i$, we see that P_i is uniformly bounded. ■

Proof of Theorem 2: By Lemma 3, the Kalman consensus error is uniformly stable. By Theorem 1, $\|\xi_i - \xi_j\| \to 0$ as $t \to \infty$ for all (i,j). Since each element of $x \to 0$ then $\|x\| \to 0$ as $t \to \infty$ and we conclude uniform asymptotic stability. Any linear system that is uniformly asymptotically stable is also uniformly exponentially stable [55]. Additionally, linear uniformly exponentially stable systems with $\|B(t)\| < \beta$ for finite β are bounded-input bounded-output stable [56]. Since the Kalman consensus error governed by Equation (3.5) is a linear uniformly asymptotically stable system with $\|B(t)\|$ bounded, it is ISS. ■

Corollary 5. *If the continuous-time consensus schemes presented in [32, 53, 34, 44] are augmented with communication noise, then the representation of these schemes that is equivalent to Equation* (3.5) *is ISS.*

Proof: The difference between each of these schemes and Equation (3.3) is that the consensus gain $K_{ij}(t)$ is time invariant. Therefore from the proof of Theorem 2 it is clear that they are ISS. ■

Referring to Figure 3.1 we see that the combination of the communication network and the consensus scheme is an ISS system. The cascade combination of two ISS systems is also ISS [57]. Therefore if the feedback loop containing the coordination

algorithm and the i^{th} vehicle is ISS from the consensus error to the cooperation objective, then the total system will be ISS from the communication noise to the cooperation objective. This concept is shown schematically in Figure 3.2 and can be summarized by the following Theorem.

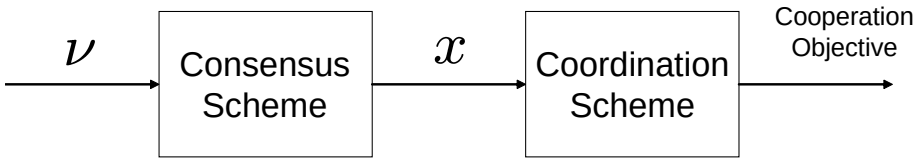


Figure 3.2: The distributed cooperative control problem can be thought of as a cascade connection between the consensus algorithm and the coordination algorithm. If both are ISS, then the cascade system will be ISS.

Theorem 6. *Consider a cascade interconnection between a coordination algorithm and a consensus scheme that is ISS from the communication noise to the consensus error. If a coordination scheme is ISS from the consensus error to the cooperation objective then the interconnected system is ISS from the communication noise to the cooperation objective.*

The major implication of Theorem 6 is that communication noise cannot permanently disrupt overall team cooperation. If a coordination algorithm is ISS and is driven by a consensus algorithm that is implemented over noisy communication channels (communication via sensing, for example), then Theorem 6 states that the error in the cooperation objective will be bounded and related to the power of the noise in the communication. When there is significant communication noise, then the cooperation objective will still be achieved, albeit loosely.

3.4 Illustrative Example - Cooperative Timing

Suppose that a team of UAVs, flying at distinct altitudes, is tasked to simultaneously visit a pre-specified location. For simplicity, also assume that paths have been precomputed for each UAV as shown in Figure 3.3.

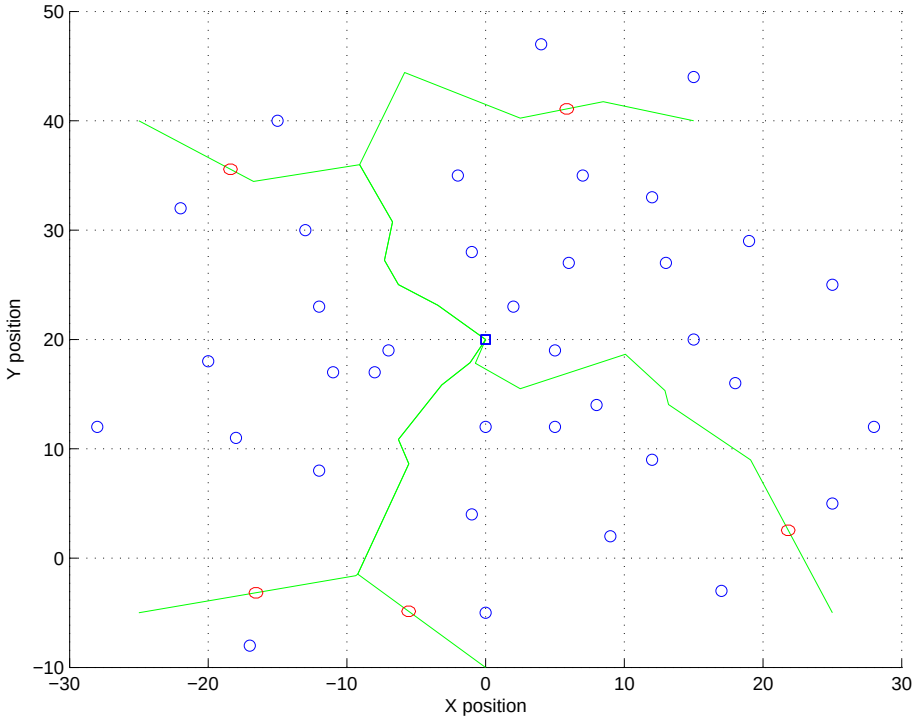


Figure 3.3: Cooperative timing scenario with five agents involved.

We will also assume that each UAV has autopilot functionality that maintains the UAV on its pre-defined path, but that the velocity along the path can be adjusted to meet the simultaneous arrival objective. We will assume that the velocity hold autopilot has been designed such that

$$\dot{v}_i = v_i^c - v_i \tag{3.6}$$

where v_i is the velocity and v_i^c the commanded velocity for the i^{th} UAV. Let L_i denote the length of the path remaining to the target, then

$$\dot{L}_i = -v_i.$$

Given L_i and v_i, the i^{th} UAV can estimate its expected time-of-arrival (ETA) as

$$\tau_i = \frac{L_i}{v_i}.$$

Differentiating, we obtain

$$\begin{aligned}\dot{\tau}_i &= \frac{v_i \dot{L}_i - L_i \dot{v}_i}{v_i^2} \\ &= -1 - \tau_i \left(\frac{v_i^c - v_i}{v_i} \right). \end{aligned} \tag{3.7}$$

The cooperation objective for this problem is that each UAV arrives at its destination simultaneously, i.e. $\tau_i - \tau_j = 0$ for each (i, j). The coordination variable for this problem is chosen as the arrival time. Therefore ξ_i represents the i^{th} UAVs understanding of the team arrival time. Clearly, to satisfy the simultaneous arrival objective, the team must come into consensus before the actual arrival time. As in many practical applications, we desire consensus in finite time, but consensus is only guaranteed asymptotically. However, due to its exponential nature, a consensus algorithm will still be useful in the presence of finite horizon requirements.

Let the commanded velocity to each UAV be

$$v_i^c = v_i + \frac{v_i}{\tau_i} \left(\gamma \tau_i - \gamma \xi_i - 1 \right), \tag{3.8}$$

then Equation (3.7) reduces to

$$\dot{\tau}_i = -\gamma \tau_i + \gamma \xi_i. \tag{3.9}$$

Note that

$$\begin{aligned}(\dot{\tau}_i - \dot{\tau}_j) &= -\gamma \tau_i + \gamma \xi_i + \gamma \tau_j - \gamma \xi_j \\ &= -\gamma \left(\tau_i - \tau_j \right) + \gamma \left(\xi_i - \xi_j \right), \end{aligned}$$

and that the system $\dot{\phi} = -\gamma \phi + \gamma u$ is input-to-state stable. In fact we have that

$$|\phi(t)| \leq e^{-\gamma (t - t_0)} \phi(t_0) + \sup_{t_0 \leq \sigma \leq t} |u(\sigma)| .$$

Therefore, the combination of the consensus strategy given by Equations (3.1)–(3.3) and the velocity controller given by Equation (3.8) is input-to-state stable with the input being communication noise and the state consisting of both the consensus discrepancy $\xi_i - \xi_j$ and the UAV arrival discrepancy $\tau_i - \tau_j$.

The cooperative timing scenario was simulated with an unreliable switching communication topology. The team is connected in the graph shown in Figure 3.4 where each link is only available 70 percent of the time. When an agent receives com-

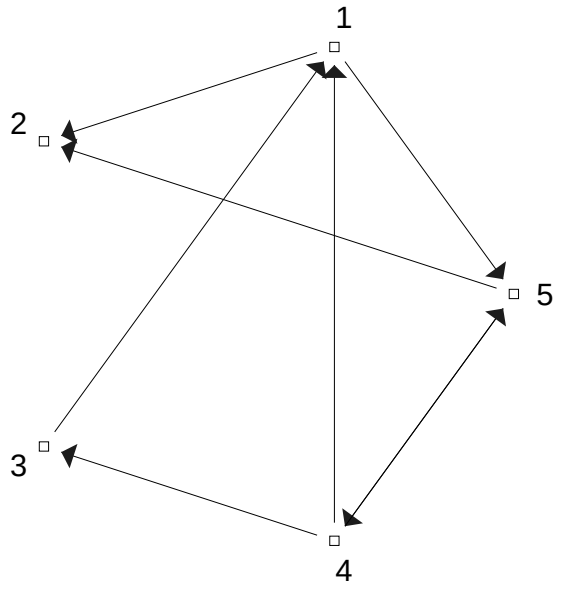


Figure 3.4: Union of possible communication topologies where each link is only available to group agents 70 percent of the time.

munication it updates its estimate of ξ_i, the team estimated time-of-arrival (ETA_{team}), using the Kalman consensus scheme of Section 3.2. In between consensus updates, agents control their velocity using Equation (3.8) so that the actual time-of-arrival matches the estimate from the consensus algorithm. Five agents were given a single target at which to arrive simultaneously, as in Figure 3.3.

In the first case, communication noise ν was set to zero and each agent started with approximately the same confidence in its estimate of the team ETA. The reference team ETA for each vehicle is shown in Figure 3.5 and the actual ETA of each

vehicle is shown in Figure 3.6. As can be seen, each agent in the team achieves agreement using consensus, adjusts its ETA to match the team ETA, and arrives at the target in approximately 20 seconds.

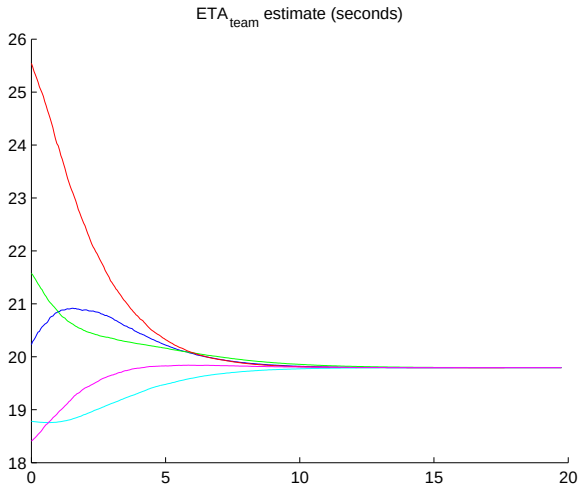


Figure 3.5: Reference team ETA for each agent with no communication noise.

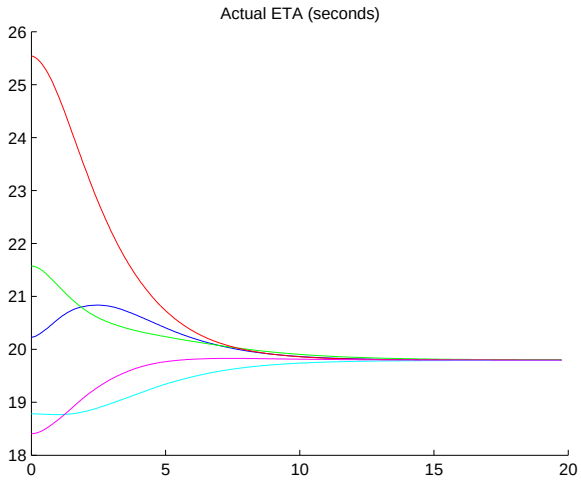


Figure 3.6: Actual ETA for each agent with no communication noise.

In the second case, significant communication noise is added. The reference team ETA for each vehicle is shown in Figure 3.7 and the actual ETA of each vehicle is shown in Figure 3.8. As can be seen, each agent in the team achieves approximate agreement using consensus where the error in agreement is due to the communication noise.

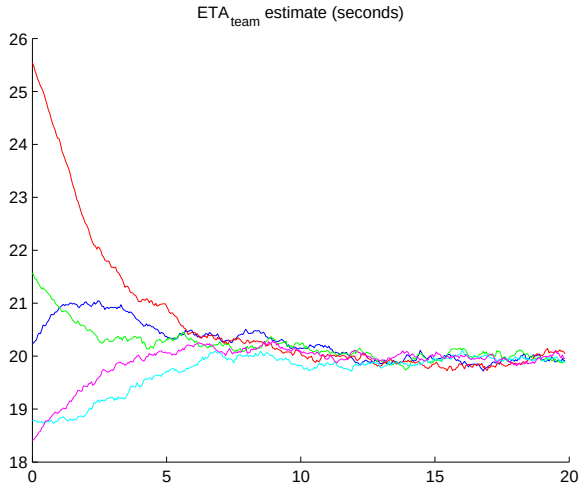


Figure 3.7: Reference team ETA for each agent with communication noise.

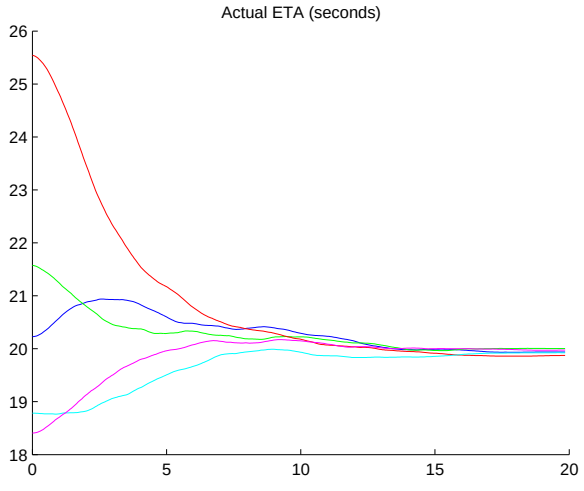


Figure 3.8: Actual ETA for each agent with communication noise.

3.5 Conclusions

This chapter has shown that the Kalman consensus scheme presented in [47] is input-to-state stable. A significant corollary is that most of the consensus schemes presented in the current literature are also ISS. The input-to-state property of the consensus scheme was used to show that if the consensus scheme is used in cascade with a multiple vehicle coordination algorithm that is also ISS, then the fidelity of the cooperation objective is directly related to the power level of the communication noise.

Chapter 4

Average Consensus and Message Passing

4.1 Introduction

For teams of autonomous agents, the ability to cooperate in a decentralized manner can enhance the overall effectiveness of the team. Central to decentralized cooperation is the consensus problem which has been investigated recently by a number of researchers [32, 33, 41, 35].

In general terms, the consensus problem for a group of agents is to ensure that as time progresses each agent approaches a consistent understanding of their shared information. In the general problem, the value to which the team converges is arbitrary, the only requirement being that all agents eventually agree. Average-consensus problems add the restriction of requiring that the final value (the group decision) be the exact average of the agents' initial values. The average-consensus problem is a part of the family of χ-consensus problems [58] where the value that the team is to converge is a function χ of the initial values of the team (e.g. max or min). Recently average-consensus has been used as a basis for distributed Kalman filters [39, 40].

In [41] Olfati-Saber and Murray propose a distributed, linear, continuous-time protocol that ensures that average-consensus is achieved asymptotically if the interaction networks connecting the agents switch between balanced, strongly connected graphs. This chapter will extend those results to the discrete-time domain as well as relax the restriction of requiring the interaction topology to be strongly connected at each instant. Our main result will be to show that if the interaction topology at any instant is balanced and the *union* of the network graph is strongly connected over every interval T, then average-consensus is still achieved asymptotically. Thus,

a network may at no instant be strongly connected, yet agents in a team can still achieve average-consensus.

This chapter is outlined as follows. In Section 4.2 we introduce a formal definition of average-consensus as well as notation that relates the communication topology to consensus protocols. Our main results are presented in Section 4.3. Section 4.4 investigates the practical issues in forming an average-consensus protocol in the discrete-time framework and proposes two such protocols. The notion of *dead-beat* consensus is introduced in Section 4.5 and Section 4.6 investigates trade-offs between asymptotic and finite-time average-consensus protocols. Finally, conclusions are offered in Section 4.7.

4.2 Definitions and Terminology

The information flow topology between agents on a team is most naturally represented as a directed graph. For this reason, we introduce graph theoretic terminology similar to [59].

Let $A = [a_{ij}]$ be an $n \times n$ nonnegative matrix. The underlying directed graph G associated with A has vertex set $V(G) = \{1, \ldots, n\}$ and a directed edge (i, j) from node i to j if and only if $a_{ji} \neq 0$ (note: some authors use the transpose of A, i.e. there is a directed edge (i, j) from i to j if and only if $a_{ij} \neq 0$). As we have defined the relationship between a matrix and its underlying graph, the nodes sending information to node i can be determined by the nonzero entries in row i. Nonzero entries in column i indicate which nodes are receiving information from node i. Note that two matrices with nonzero entries in the same locations have the same underlying graph. The neighbors, N_i, of node i are all nodes that communicate to i, i.e. $N_i = \{j \mid a_{ij} \neq 0\}$. By convention, we assume that each node can communicate with itself, so $a_{ii} > 0 \ \forall i$ and $i \in N_i$.

The graphs associated with matrices can be connected in a variety of ways. Connectivity of the network can be roughly classified as follows:

- *Fully Connected:* Each node has as its neighbors all other nodes in the network.

- *Strongly Connected:* Each node has a path that follows the directed edges of the graph to every other node in the network. A direct connection to all other nodes is not necessary, but information flow from each node must reach all other nodes.
- *Spanning Tree:* At least one node has a path that follows the directed edges of the graph to every other node in the network.

Graphs can also be connected over time by considering the union of the communication links over an interval of time (i.e. the union contains all edges that were active during that interval). A reversed graph is simply a graph with the direction of the links reversed. Note that a reversed graph is associated with the transpose of the original matrix.

Each node has an associated value $x_i \in \mathbb{R}$ which represents the information on which the team must come to agreement. The set of nodes $\{1, \ldots, n\}$ is said to be in consensus if $x_i = x_j$ for all i, j. When each $x_i = \frac{1}{n} \sum_j x_j[0]$ the team is said to have reached average-consensus. A consensus protocol defines how a node should update is value of x_i based on the values of its neighbors. The simplest scheme is to require that each node update its value x_i to some weighted linear combination of its neighbors values

$$x_i[k{+}1] = \sum_{j \in N_i} a_{ij} x_j[k].$$

The dynamics of the information vector $x = \{x_1, \ldots, x_n\}$ can then be defined as

$$x[k{+}1] = A[k]x[k]$$

where the *sign* of each entry in $A[k]$ is given by the communication topology at time k, but the value a_{ij} for the nonzero elements is determined by the protocol.

Let $\Phi_A(k, k_0) = A[k]A[k{-}1] \cdots A[k_0]$, then at each k the information vector can be described by

$$x[k{+}1] = \Phi_A(k, 0)x[0].$$

Consensus is said to be reached asymptotically if

$$\lim_{k\to\infty} \Phi_A(k,0) = \mathbf{1}y^T \tag{4.1}$$

where $\mathbf{1}$ is the vector of all ones, $y_i \geq 0$, and $\mathbf{1}^T y = 1$. Notice that if Eq. (4.1) is satisfied, then $x \to \mathbf{1}y^T x[0]$ implying that each x_i approaches the same convex combination of the agents' initial values. Equivalently, average-consensus is said to be reached asymptotically if

$$\lim_{k\to\infty} \Phi_A(k,0) = \frac{1}{n}\mathbf{1}\mathbf{1}^T. \tag{4.2}$$

4.3 Average-Consensus under Switching Topologies

The results for linear consensus protocols under switching interaction topologies have been well studied [33, 35] with the main result being that the union of the interaction graphs over every interval T must contain a spanning tree to reach consensus. We will draw similar conclusions with respect to average-consensus. Theorem 1 develops the conditions for each $A[k]$ that allows Eq. (4.2) to be satisfied. This requires the following two Lemmata.

Lemma 1 (Proposition 1 in [35]). *Let $x[k+1] = A[k]x[k]$ where $A[k] = [a_{ij} \geq 0]$, $\sum_j a_{ij} = 1$, $a_{ii} > 0$ for all k, and each nonzero entry a_{ij} is both uniformly upper and lower bounded. If there exists $T \geq 0$ such that for every interval $[k, k+T]$ the union of the interaction graph across the interval contains a spanning tree, then consensus is asymptotically achieved (i.e. Eq. (4.1) is satisfied).*

A similar result is implicit in [47]. Notice that each node has the ability to choose the weight associated with the information from each of its neighbors to ensure that its row sums to one. If the team is connected often enough (i.e. has a spanning tree over every interval of length T), then Lemma 1 ensures that consensus is reached.

Lemma 1 requires that the row sums of $A[k]$ be one and that a spanning tree be achieved in every interval of length T for consensus to be reached. Now consider the reversed dynamics $x[k+1] = B[k]x[k]$ where each *column* sum is equal to one.

Lemma 2. *Let* $x[k+1] = B[k]x[k]$ *where* $B[k] = [b_{ij} \geq 0]$, $\sum_i b_{ij} = 1$, $b_{ii} > 0$, *and each nonzero entry* b_{ij} *is both uniformly upper and lower bounded. Under switching interaction topologies, if there exists* $T \geq 0$ *such that for every interval* $[k, k+T]$ *the union of the* reverse *interaction graph across the interval contains a spanning tree, then*

$$\lim_{k\to\infty} \Phi_B(k, k_0) = y\mathbf{1}^T$$

where $y_i \geq 0$ *and* $y^T\mathbf{1} = 1$.

Proof: If the column sums of $B[k]$ are equal to one and a spanning tree is achieved in the reverse graph, then $B^T[k]$ has row sums of one and a spanning tree is achieved in the regular graph. By application of Lemma 1 $\lim_{k\to\infty} \Phi_{B^T}(k, k_0) = \mathbf{1}z^T$, so

$$\begin{aligned}
\lim_{k\to\infty} \Phi_{B^T}(k_0, k) &= \mathbf{1}y^T \\
\left[\lim_{k\to\infty} \Phi_{B^T}(k_0, k)\right]^T &= \left[\mathbf{1}y^T\right]^T \\
\lim_{k\to\infty} \Phi_{B^T}(k_0, k)^T &= y\mathbf{1}^T \\
\lim_{k\to\infty} \Phi_B(k, k_0) &= y\mathbf{1}^T \ .
\end{aligned}$$

The fact that $\Phi_{B^T}(k, k_0) = \mathbf{1}z^T \Rightarrow \Phi_{B^T}(k_0, k) = \mathbf{1}y^T$ can be seen by noting that each $B^T[k]$ is row stochastic with positive diagonal entries and if the product $B^T[k]B^T[k+1]$ $\cdots$ $B^T[k+T]$ contains a spanning tree, then the product $B^T[k+T]B^T[k+T-1]$ $\cdots$ $B^T[k]$ also contains a spanning tree. Wolfowitz [54] showed that infinite products of SIA matrices (a superset of matrices that have a spanning tree and are row stochastic with positive diagonal entries) converge to the form $\mathbf{1}y^T$ in *any* product order (however, the value of y will be dependent on the actual order). ■

Theorem 1. *Let* $x[k+1] = A[k]x[k]$ *where* $A[k] = [a_{ij} \geq 0]$, $\sum_i a_{ij} = 1$, $\sum_j a_{ij} = 1$, $a_{ii} > 0$, *and each nonzero entry* a_{ij} *is both uniformly upper and lower bounded. Under switching interaction topologies, if there exists* $T \geq 0$ *such that for every interval* $[k, k+T]$ *the union of the interaction graph across the interval is strongly connected, then Eq.* (4.2) *is satisfied and average-consensus is reached asymptotically.*

Proof: Since $A[k]$ is strongly connected over each interval $[k, k+T]$, then $A[k]$ has a spanning tree in both the regular graph and the reverse graph. Therefore, the matrix

$A[k]$ satisfies all the conditions in Lemma 1 *and* Lemma 2. Consequently,

$$\lim_{k\to\infty} \Phi_A(k, k_0) = \mathbf{1}y^T$$

and

$$\lim_{k\to\infty} \Phi_A(k, k_0) = z\mathbf{1}^T$$

so

$$\lim_{k\to\infty} \Phi_A(k, k_0) = \frac{1}{n}\mathbf{1}\mathbf{1}^T.$$

■

4.4 Distributed Protocol

Careful examination of Theorem 1 will reveal that finding a distributed protocol to satisfy the hypotheses of the theorem will be difficult. Specifically, at each instant in time, the row *and* column sums must be equal to one. In the general consensus problem, only the row sums are required to be one. Since the neighbors of agent i are determined completely by row i, then each agent simply chooses appropriate weights for each of its neighbors values ensuring that the weights sum to one. In the average-consensus case, not only do the weights associated with the neighbors of i need to sum to one, but all nodes for which i is a neighbor must weight the information from i such that the column sum is equal to one. This section will investigate this subtlety and propose two protocols that achieve average-consensus in a distributed manner.

To illustrate the difficulty of requiring both row and column sums to be one, consider the network topology shown in Figure 4.1. The matrix

$$A = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 0 & 2 & 1 \\ 2 & 0 & 1 \end{bmatrix}$$

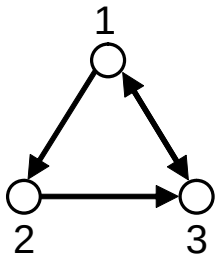


Figure 4.1: Simple network over which the average-consensus problem can be solved, but which requires global information to be available.

has as its underlying graph the topology shown in Figure 4.1 and has row and column sums equal to one. If the underlying network topology remains fixed, then by Theorem 1, the system will achieve average-consensus asymptotically. In one sense, the protocol is distributed since each agent only uses the information received from its neighbors; however, the first agent weights all neighbors' values equally, the second agent weights its own value twice as much as its neighbors, and the third agent weights its neighbors values twice as much as its own. In order to determine the entries in A, some global knowledge of the network topology is required — i.e. there is no simple rule that an agent can use to determine the weight it gives to information from its neighbors without knowledge of the global topology.

An ideal protocol would be able to achieve average-consensus without using global information. We will investigate two protocols that impose additional restrictions on the types of graphs involved, but that achieve average-consensus without resorting to global information. The first is proposed in [41] and requires the definition of the graph Laplacian. Let L be defined element-wise as

$$\ell_{ij} = \begin{cases} \sum_{k=1,k\neq i}^{n} \alpha_{ik}, & j = i \\ -\alpha_{ij}, & j \neq i \end{cases}$$

where $\alpha_{ij} = 1$ if there is a communication link from node j to node i and $\alpha_{ij} = 0$ otherwise (here $\mathcal{A} = [\alpha_{ij}]$ is simply the adjacency matrix of a graph G). The protocol

is then defined in terms of the Laplacian

$$x[k+1] = (I - \epsilon L)x[k] \tag{4.3}$$

where $\epsilon \in (0, 1/\max_i \ell_{ii})$. Notice that the row sums of L are all zero by construction, so the row sums of $A = I - \epsilon L$ are all one. If $\epsilon \in (0, 1/\max_i \ell_{ii})$, then A will also be nonnegative and consensus will be guaranteed if the graph contains a spanning tree in every interval T.

Olfati-Saber and Murray show in [41] that when a graph is *balanced* then the column sums of L are zero. A balanced graph is one in which at each node the out-degree equals the in-degree, i.e. each node sends information to as many as send information to it. Notice that when G is balanced then L has column sums of zero, and A has column sums of one. So, by Theorem 1, the protocol (4.3) will achieve average-consensus if the network switches between instantaneously balanced networks which are strongly connected over every interval T.

Protocol (4.3) is almost completely distributed since each node determines the weight to associate with information from its neighbors without knowledge of the graph topology; however, all nodes must have the same value of ϵ whose upper limit is determined by the connectivity of the graph. Certainly, for a fixed number of agents n, $\epsilon \in (0, 1/n]$ will ensure that A remain nonnegative and Theorem 1 will apply. This requires *a priori* knowledge of the size of the team, especially since the larger the value of ϵ the faster the rate of convergence (the second eigenvalue will be closer to zero, see [41]). Setting $\epsilon = 1/N$ where N is an upper bound on the number of agents on the team will ensure that average-consensus is achieved asymptotically.

To illustrate the applicability of this protocol consider a scenario where the network topology switches randomly from between the graphs in Figure 4.2. The convergence for two values of ϵ are shown in Figures 4.3 and 4.4. In this example, the sum of the initial conditions is one. Notice that with both values of ϵ the value to which the system converges is $\frac{1}{4}$, but the larger value of ϵ gives faster convergence.

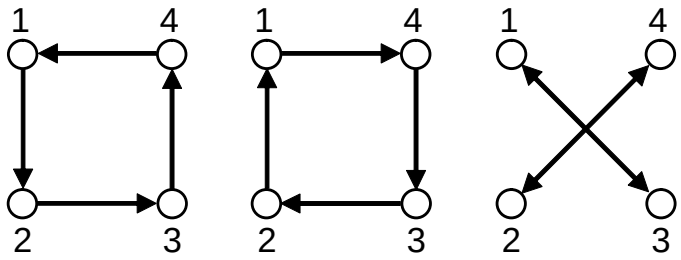


Figure 4.2: Example scenario where the network topology switches randomly between these three graphs.

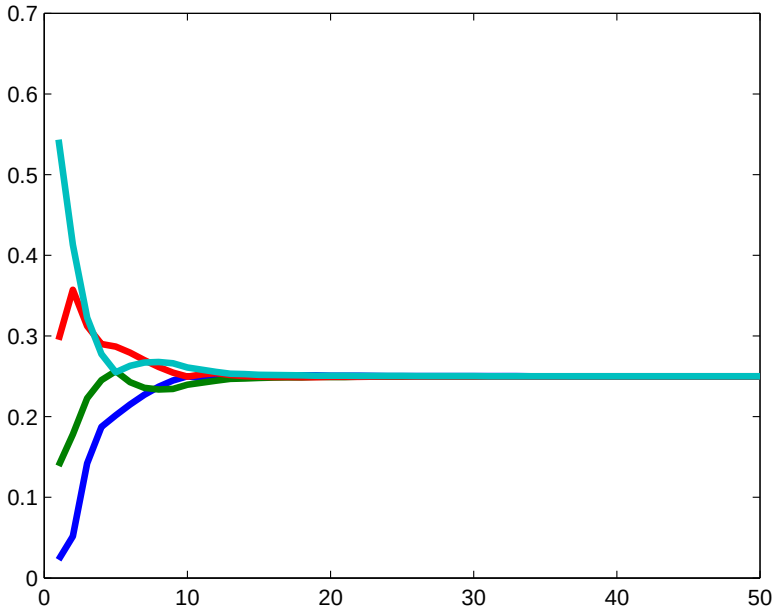


Figure 4.3: Protocol (4.3) with $\epsilon = \frac{1}{4}$ under switching topologies.

When every communication link is bi-directional (i.e. G is an undirected graph), then the graph is trivially balanced. In this case, it is possible to develop a protocol that can be implemented without *a priori* knowledge of team size. Assume that agents have the ability to negotiate with each of their neighbors to isolate the exchange of information to just one neighbor at a time. During this communication event, both agents update their values to be the exact average of the values present. For a two-agent communication event (i, j), the protocol matrix A will be the identity matrix with the exception of $a_{ij} = a_{ji} = a_{ii} = a_{jj} = \frac{1}{2}$. Notice that each $A[k]$ retains

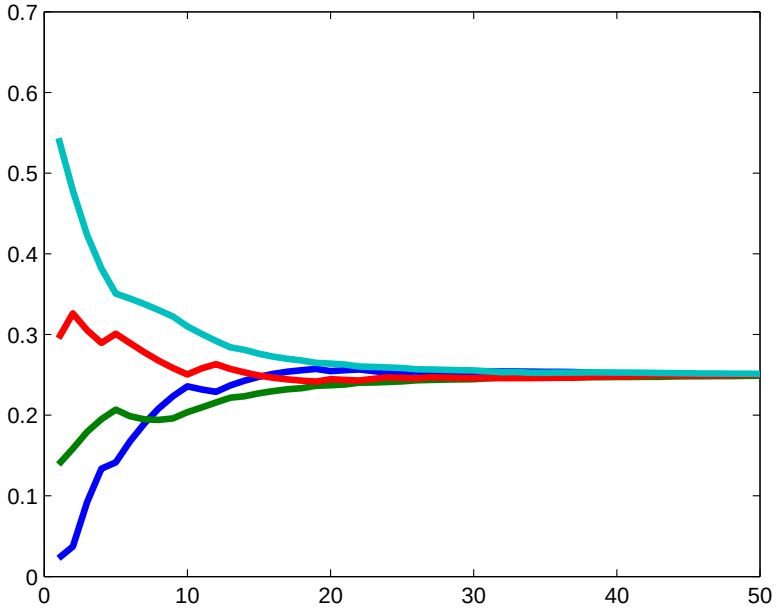


Figure 4.4: Protocol (4.3) with $\epsilon = \frac{1}{8}$ under switching topologies.

the characteristic of having row and column sums equal to one. Essentially, each agent cycles through available communication channels to isolate a single neighbor at a time and effectively change its in-degree and out-degree to one at each instant. If over every interval T the union of these simple graphs is connected, then the conditions in Theorem 1 are satisfied and average-consensus is achieved asymptotically.

An example of this protocol is shown in Figure 4.6. In this scenario, the agents are connected in a static graph of the form $\{1 \leftrightarrow 2, 2 \leftrightarrow 3, 3 \leftrightarrow 4\}$. The agents negotiate with their neighbors so that each agent only communicates with one other agent at a time. For simulation purposes, this can be modeled as the system switching randomly between the graphs in Figure 4.5. Observe that the final value is the exact average of the agents' initial conditions.

To summarize, the Laplacian protocol of Eq. (4.3) can achieve average-consensus if the interaction topology is balanced at each instant and is strongly connected over every interval T. It requires that some knowledge of the maximum connectedness or maximum number of agents be available *a priori* to determine the parameter ϵ.

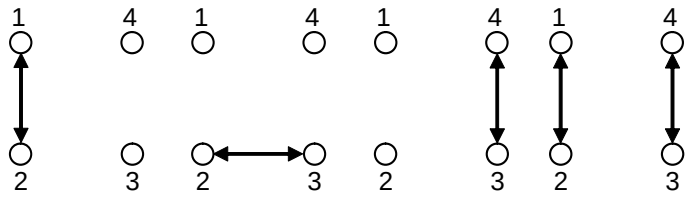


Figure 4.5: Example scenario where the topology remains fixed (a path), but the agents negotiate through one of the above graphs at each instant.

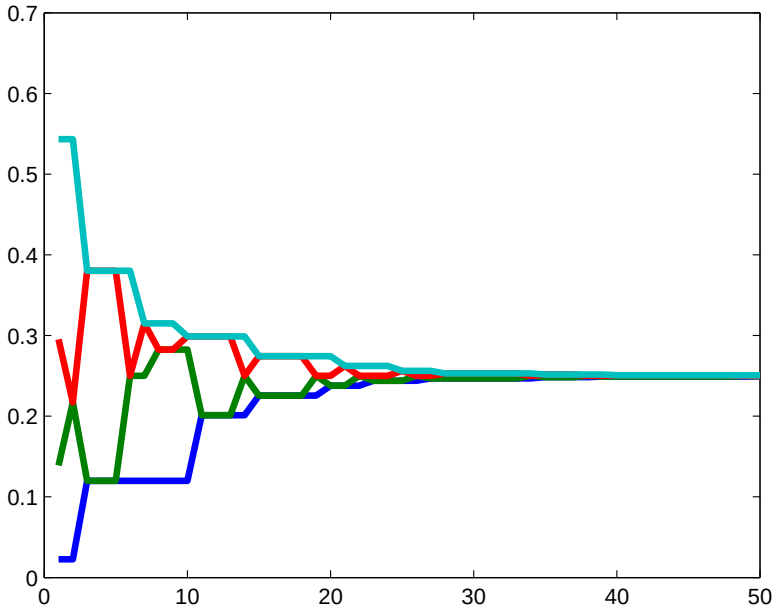


Figure 4.6: Results of simple two-agent events to achieve average consensus.

A second protocol reduces available communication to allow only simple two-agent interactions. This requires bi-directional communication between agents (more restrictive than balanced) but assumes no *a priori* knowledge of the network topologies or team size.

4.5 Deadbeat Consensus

Discrete-time systems can exhibit finite-time convergence when the poles of the system are all at zero — i.e. when a system

$$x[k+1] = Ax[k]$$

has nilpotent matrix A, then $x[k] = 0\ \forall k > \dim(A)$, regardless of initial conditions. This notion of "deadbeat" response motivates a similar investigation of consensus systems. This section will consider the conditions under which consensus can be reached in finite time.

Let $\mathcal{P}$ be a consensus protocol; specifically, let the interaction matrices generated by $\mathcal{P}$ have row sums equal to one. Note that if $\mathcal{P}$ solves either the general consensus problem or the average-consensus problem, the row sums of the interaction matrices will be one. At any time k, the value of the system given the initial conditions at $k = 0$ is

$$x[k{+}1] = (A[k]A[k{-}1]\cdots A[1]A[0])x[0].$$

Theorem 2. *Let $\mathcal{P}$ be a consensus protocol. If at some instant, ℓ, the interaction topology is a fully connected graph and $\mathcal{P}$ yields the interaction matrix at that instant*

$$A[\ell] = \frac{1}{n}\mathbf{1}\mathbf{1}^T$$

then the team will be exactly in consensus for all $k > \ell$.

Proof: A group of agents is in consensus if $x_i = x_j$ for every pair (i, j). Since

$$x[\ell{+}1] = A[\ell]x[\ell] = \frac{1}{n}\mathbf{1}\mathbf{1}^T x[\ell]$$

then each element i of $x[\ell{+}1]$ is

$$x_i[\ell{+}1] = \frac{1}{n}\sum_{j=1}^{n} x_j[\ell].$$

Therefore, the group has reached consensus at time $(\ell{+}1)$. Because $\mathcal{P}$ ensures that each interaction matrix after time ℓ has row sums equal to one, then for all $k > \ell$ the group remains in consensus (each node updates to a weighted sum of the same value).

To show that the same conditions on A lead to deadbeat average-consensus, notice that $\mathcal{P}$ has row and column sums equal to one at each instant so

$$\sum_{i=1}^{n} x_i[k] = \sum_{i=1}^{n} x_i[0]$$

for all $k \geq 0$. Recall that if a matrix A has row and column sums equal to one, then $\mathbf{1}$ is both a left and right eigenvector associated with eigenvalue 1, so

$$\begin{aligned}
\sum_{i=1}^{n} x_i[k] &= \mathbf{1}^T x[k] \\
&= \mathbf{1}^T A[k] x[k-1] \\
&= \mathbf{1}^T x[k-1] \\
&\vdots \\
&= \mathbf{1}^T x[0] \\
&= \sum_{i=1}^{n} x_i[0].
\end{aligned}$$

Therefore, for each agent i at time ℓ

$$x_i[\ell+1] = \frac{1}{n} \sum_{j=1}^{n} x_j[\ell] = \frac{1}{n} \sum_{j=1}^{n} x_j[0].$$

which implies that the group has reached average-consensus. ■

Theorem 2 requires that at some instant the communication graph is fully connected, i.e every agent can communicate with every other agent *and* that the interaction matrix generated by the consensus protocol yields $A = \frac{1}{n}\mathbf{1}\mathbf{1}^T$. One consequence of this is that deadbeat average-consensus is much more difficult to achieve than regular deadbeat consensus. This is due to the fact that average-consensus protocols do not generally yield the proper interaction matrix when the communication graph is fully connected.

The reader will note that even when a graph is fully connected neither protocol from Section 4.4 will yield the proper interaction matrix to achieve deadbeat

consensus. A consensus protocol of the form

$$x_i[k+1] = \frac{1}{|N_i|} \sum_{j \in N_i} x_j[k] \tag{4.4}$$

will allow regular deadbeat consensus since whenever the network is fully connected, each agent updates its value to the average of all the other agents. Unfortunately, such a simple protocol does not lead to average-consensus in the general case. Consider the balanced network shown in Figure 4.7 with interaction matrix

$$A = \frac{1}{6} \begin{bmatrix} 3 & 0 & 3 & 0 \\ 2 & 2 & 0 & 2 \\ 0 & 3 & 3 & 0 \\ 0 & 3 & 0 & 3 \end{bmatrix}$$

which does not have column sums equal to one. In fact

$$\lim_{k \to \infty} A^k = \frac{1}{9} \begin{bmatrix} 2 & 3 & 2 & 2 \\ 2 & 3 & 2 & 2 \\ 2 & 3 & 2 & 2 \\ 2 & 3 & 2 & 2 \end{bmatrix}$$

which shows that the protocol defined by Eq. (4.4) is not an average-consensus protocol. It is interesting to note that if the network topologies switch between balanced

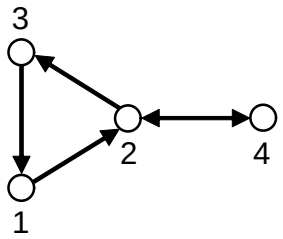


Figure 4.7: Balanced graph for which a simple averaging protocol does not achieve average-consensus.

regular graphs (graphs where all nodes that have any adjacent edge have the same degree), then this protocol does achieve average-consensus (in the general and deadbeat case).

4.5.1 Example Application

Consider a fixed perimeter which is to be monitored by a team of N agents in a distributed manner (as in [60]). Let the consensus variable in the system be the size of the segment an agent is to monitor. The initial state of the system is when the first agent reaches the endpoint of the perimeter and initializes its consensus variable to the length of the perimeter. We desire average-consensus so that asymptotically, each agent monitors an equal part of the perimeter.

Using the protocol of Eq. (4.4) and noticing that the system will only switch between balanced regular graphs (since agents meet along a line) deadbeat average-consensus may be reached. Figure 4.8 shows a scenario where at no time are all agents in communication, so average-consensus is achieved asymptotically. In contrast, Figure 4.9 shows a scenario where agents are launched in close proximity and meet in a fully connected group near the beginning of the mission achieving deadbeat average-consensus.

4.6 Finite-Time Average-Consensus

An average-consensus protocol will invariably require a strongly connected network since each agent must be able to influence the group decision to reflect its initial condition. Obviously, if each agent transmits its initial condition to its neighbors and passes any communication received from others along, then if a strongly connected network is available, eventually all agents will have the complete set of initial conditions from which the average can be computed (clearly, this is not novel; Lynch [29] classifies such an algorithm as trivial). Using this method, all agents will have the information necessary to be in average-consensus after d steps where d is the diameter of the graph. Indeed, it may seem that the restriction to a strongly connected network eliminates any need for an asymptotic protocol. This section will

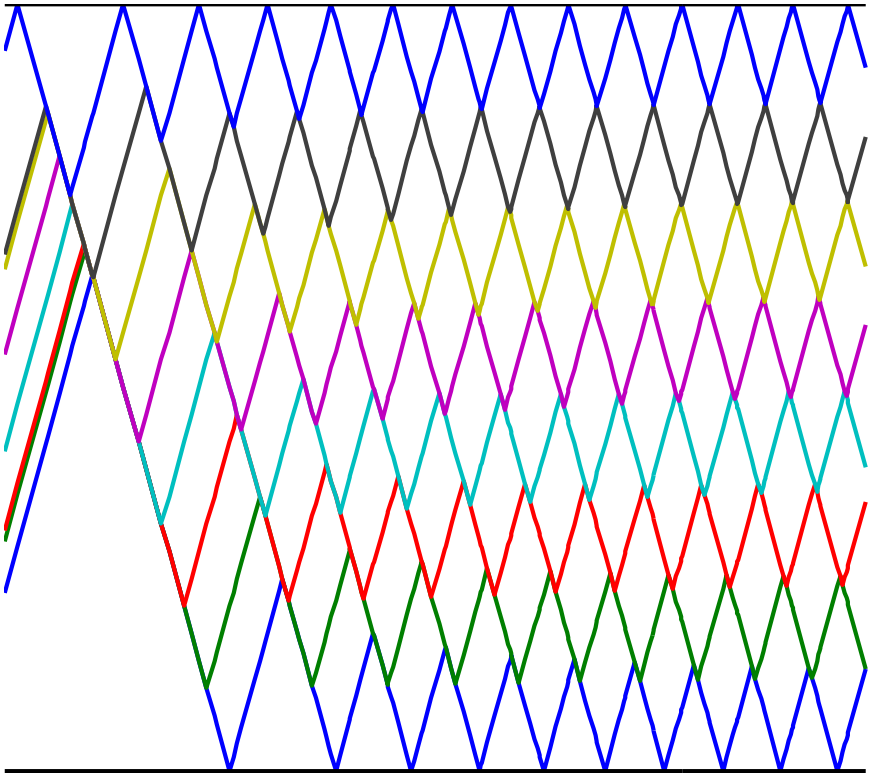


Figure 4.8: Perimeter surveillance using average-consensus to distribute the team evenly along the perimeter.

investigate the trade-offs between an asymptotic protocol (such as (4.3)) and a simple message passing protocol.

The main advantage of an asymptotic consensus protocol is the small amount of bandwidth required — each agent needs only to send its current value. Additionally, there is no need to identify individual agents or know the number of agents in the team. On the other hand, a message passing protocol could keep track of which initial conditions it has sent to each of its neighbors and effectively limit its bandwidth to be the same as the asymptotic protocol, relying instead on repeated interaction to transmit all initial conditions. At each instant, a node's value would be the sample average, i.e. the average of all initial conditions received so far. For large networks, however, the amount of overhead and the complexity may be prohibitive. Perhaps the main advantage of the message passing protocol is the ability to utilize any type

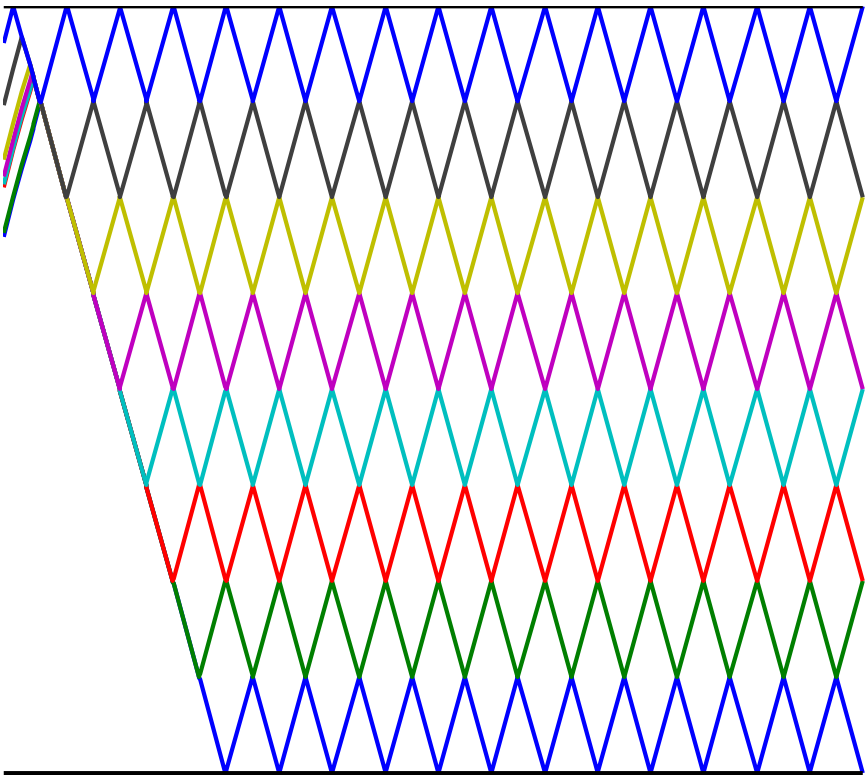


Figure 4.9: Perimeter surveillance where deadbeat average-consensus occurs.

of data (not simply continuous real numbers) in any functional way, i.e. agents are not limited to average-consensus but they can come to agreement on any function of the initial conditions.

The message passing protocol effectively emulates a fully connected graph at $k = 0$ by transmitting the required information incrementally. The deadbeat nature of the protocol makes it attractive, especially when speed of convergence is an issue.

An asymptotic protocol will be useful in very large networks and in situations when the value at each node is driven by an external source (agent values are dynamic rather than static) such as distributed Kalman filtering [39]. In many cases, however, a simple message passing scheme may be more attractive due to its deadbeat nature and its ability to handle any data type.

In an effort to quantify the performance of average-consensus as compared to a message passing protocol, we performed a number of Monte-Carlo simulations. We

varied team size and available bandwidth to determine under which circumstances and how quickly the average consensus and message passing algorithms would converge. For different bandwidth sizes, the message passing protocol picked a random set of the bandwidth size and passed those values to its neighbors at every time step. For example, in the bandwidth size 2 case, the message passing protocol would pick 2 random values from its vector of known team values and pass those to its neighbors. In the N bandwidth case, an agent would send all values that it has seen to its neighbors. In every bandwidth case, the average-consensus protocol simply took the average of its immediate neighbors and passed only one message to each of its neighbors.

Each simulation consisted of 2000 iterations with a fixed team size and bandwidth. Table 4.1 shows the results of the tests where each element of the table corresponds to a specific team size/bandwidth test. Each cell contains the mean and standard deviation for the number of iterations required to converge to within a small amount of the final consensus value. The top numbers in a cell correspond to the message passing protocol and the colored numbers correspond to the average consensus algorithm. Note that for a fixed team size, the consensus algorithm performs the same across all bandwidths. Since the consensus algorithm only passes its current value to its neighbors, it is unable to exploit the additional bandwidth.

As can be seen in Table 4.1, the message passing protocol outperforms the average consensus when there is unlimited bandwidth (i.e. bandwidth size N). In that case, every agent communicates to its neighbors all the initial conditions of the team that it has seen previously. Even for a large team size of 64 agents, the message passing protocol converges an order of magnitude faster than the average-consensus algorithm. As the available bandwidth goes down, however, the average-consensus algorithm shines. Even with a small team size of 16 agents, at a bandwidth of 1 message the average-consensus algorithm converges twice as fast as the message passing protocol. We postulate that more information is contained in the value transmitted by the consensus algorithm since it is a weighted average of its neighbors values. As

bandwidth increases, however, the message passing algorithm is able to increase the information transmission to its neighbors.

We conclude that for bandwidths much less than the size of the team, the average-consensus algorithm will on average converge faster than a message passing protocol. When there is an abundance of available bandwidth on the order of the team size, then the message passing protocol will be the best choice.

Table 4.1: Average Iterations to Consensus

Team Size	Bandwidth: 1	2	4	$\frac{N}{2}$	N
4	31.69 (17.95)	15.83 (8.348)	9.078 (4.011)	15.98 (8.371)	8.989 (3.92)
	109.9 (17.62)	109.7 (17.83)	109.7 (18.38)	111.2 (18.2)	110.3 (18.24)
8	190.2 (91.81)	94.1 (45.86)	47.58 (23.05)	46.58 (22.1)	27.95 (12.23)
	333.4 (44.37)	333.5 (44.77)	333.6 (44.51)	331.4 (43.28)	334.3 (46.01)
16	1052 (350.7)	542.9 (234.9)	272.3 (120.8)	134.4 (59.23)	75.5 (30.2)
	837.4 (115.3)	837.9 (114)	841.8 (115.1)	837.9 (116.5)	839.4 (114.3)
32	3166 (148.6)	2545 (625.9)	1393 (556.8)	340.9 (145.4)	186.6 (67.65)
	1862 (269.9)	1857 (263.4)	1853 (260.4)	1857 (268.7)	1856 (264.2)
64	6400 (0)	6388 (95.83)	5447 (1042)	721.2 (283.3)	410.6 (131.4)
	3796 (517.8)	3824 (526.9)	3807 (526.1)	3811 (506.2)	3774 (515.3)

In practical scenarios with wireless radio modems, it will often be the case that the overhead required to begin communication with an agent encourages larger communication packets (due to the cost of switching between neighbors). If the team is known to be moderate sized, the savings in convergence time may be substantial by using a message passing protocol. Finally, we reiterate that consensus algorithms can only come into agreement on continuous valued variables; a message passing scheme allows any function of the initial conditions to be used, including discrete functions such as voting protocols. For very limited communication bandwidth or very large teams, consensus algorithms are a good choice.

4.7 Conclusions

This chapter has extended the average-consensus results of [41] to allow for networks to switch between instantaneously balanced networks that are strongly connected over every interval T. The discrete-time case has been dealt with explicitly and two asymptotic protocols presented that achieve average-consensus under switching topologies. The notion of deadbeat consensus was investigated with conclusion that general consensus problems may best be solved using a message passing mechanism rather than defining dynamics of the information variable if a strongly connected network can be assumed.

Chapter 5

Circle Surveillance

5.1 Introduction

A primary use of unmanned air vehicle (UAV) systems is in surveillance and reconnaissance missions [61, 62]. We investigate the use of a team of multiple UAVs orbiting a target with application to target tracking and convoy support.

The payload of choice for most small UAVs is a camera. The objective of our work is to develop a cooperative guidance strategy to distribute UAV agents around an orbit spaced equally in angle. The equal angle spacing allows the team to cooperatively overcome possible line-of-sight occlusions, i.e. equal spacing gives the team the best chance to track a target in the presence of occlusions. We note that for two UAVs carrying radar sensors, line-of-sight angles separated by 90 degrees provide better statistical performance in the tracking problem [63] and when the team size is greater than two, equal spacing has good performance. In a general surveillance mission, the equal spacing of the sensors provides the best overall coverage of a target and its surroundings.

The design of a spacing controller is strongly influenced by the capabilities of the UAVs on the team. For instance, helicopters can hover at a specific location and thereby maintain persistant coverage of a ground based target, however fixed-wing aircraft must fly above the stall velocity, and may therefore not be able to maintain persistent coverage. Furthermore, fixed-wing aircraft fly most efficiently at a fixed, nominal airspeed. One approach to equal spacing is to adjust the local velocity of the agents along the desired orbit. However, for small allowable velocity bounds, the convergence to the equilibrium configuration may be sluggish. Additionally, maintaining fixed-wing aircraft at their constant fuel efficient velocity is desirable from a mission

duration standpoint. In this chapter we develop a spacing controller that steers the UAVs to the desired configuration while holding a constant airspeed.

Other researchers have studied the problem of spacing fixed-speed UAVs around a possibly moving target. Paley et al. introduce the notion of the splay state configuration and give an elegant control solution for fixed target problems [64]. Their approach relies on invariant set arguments to show that the splay state configuration is the stable equilibrium of the system. The main drawback of their work is the inability to specify the orbit center. The splay state configuration is shown to be stable around the collective center of mass *not* a specific target location which makes tracking a moving target infeasible without modifications. Additionally, the control signal exhibits slow transient response for large initial errors.

Paley's splay state configuration work is extended by Klein and Morgansen in [65] to moving targets. By choosing a control signal that preserves the invariant sets introduced by Paley, they are able to design an algorithm to track a moving target in the splay state configuration with 3 UAVs. Unfortunately, the method does not currently extend to team sizes other than $N = 3$.

Frew and Lawrence [61] use vector field notions to steer a team of two UAVs to an orbit centered on a moving target. A limit cycle is designed as the equilibrium of the vector field dynamics and is modified to account for spacing errors. No formal proof is offered in their method and only team sizes of $N = 2$ are considered.

The unique features of our approach are the ability to include an arbitrary number of team members in a moving target scenario and the determination of bounds on target velocity for which the algorithm satisfies the UAV's kinematic constraints. Additionally, the transient response is qualitatively better than other approaches. Of note is that our algorithm is completely decentralized where agents base their actions only on communication from immediate team members. This allows for dynamic changes to the team to be accounted for without global communication or replanning. A drawback to our approach is that global stability is not conclusively shown, although Monte-Carlo simulations indicate that the splay state configuration is the globally stable equilibrium of the system.

The aim of this chapter is to present a stable, decentralized spacing controller for fixed-velocity UAVs tracking moving targets in the presence of wind. Section 5.2 formally defines the notion of equal spacing and describes the mathematical model that we use for the UAVs. Section 5.3 establishes the heading design for a group of UAVs monitoring a stationary target. In Section 5.4, we analyze the stability of the system for the stationary target case. These results are extended to the moving target/wind case in Section 5.5 and we conclude with simulation and hardware results in Section 5.6. Concluding remarks are offered in Section 5.7.

5.2 Problem Description

In a variety of applications the ability for a team of UAVs to spread out in some manner increases the efficiency of the team as a whole. For single target surveillance, a team of UAVs spaced equally around an orbit centered on the target gives the best line-of-sight coverage in the presence of occlusions. This chapter focuses on constructing a desired heading for each UAV in the team to achieve equal spacing. The desired heading is calculated based on the distance away from the desired orbit and the spacing error from the splay state configuration.

Definition 1 (Splay State Configuration)**.** *A set of agents $\mathcal{I}$, all of which are following the same periodic trajectory, is said to have reached the* splay state configuration *if for each agent i, the time difference of arrival to a specific point on the trajectory between agent i and its two immediate neighbors is constant for all $i \in \mathcal{I}$.*

Definition 1 describes the splay state configuration as equally spaced in time along a periodic trajectory. When agents pass a reference point (arbitrarily chosen) on the trajectory at equal time intervals, the team has reached the splay state configuration. For simple circular trajectories, the splay state configuration is achieved when agents are equally spaced in angle around the circle perimeter. Note that equal angular spacing matches the definition of the splay state configuration in [64]. Definition 1 extends the splay state notion to non-circular trajectories which occur when the center of the desired orbit is changing in time due to wind or target motion.

Consider a circular trajectory with all agents traveling at constant speed V. The time difference of arrival corresponds to the angle separation between neighbors. When the angle between all agents is the same then the splay state configuration has been reached, i.e. the agents are equally spaced in angle around the circle. Now consider the trajectory shown in Figure 5.1, which is an example of a UAV orbiting a moving target. Note that as the target speed increases, the ability for the UAV to maintain an orbit around the target depends on its ability to make increasingly sharp turns. Constraints on the turning radius of the UAV will lead to a threshold value of target speed where feasible tracking is no longer possible (see Section 5.5). In a

Figure 5.1: For a UAV orbiting a moving target, the trajectory exhibits loops corresponding to the times when the UAV and the target are moving in opposite directions and long arcs when both are moving in the same direction.

moving reference frame (with the target in the center) the motion of the UAV traces out a circle, but the splay state configuration does *not* correspond to equal spacing in angle around that circle. Since the target is moving, a much greater amount of time is spent on the part of the trajectory where the UAV and the target are moving in the same direction. When the target and UAV are moving in opposite directions, the UAV quickly travels around a large portion of the circle. Figure 5.2 shows the splay state configuration for 5 UAVs when the target is moving at 75% of V in zero wind conditions.

5.2.1 UAV Modeling

To maximize fuel efficiency each UAV maintains a constant airspeed. Additionally, we assume that all UAVs fly at a fixed altitude. A kinematic model for a

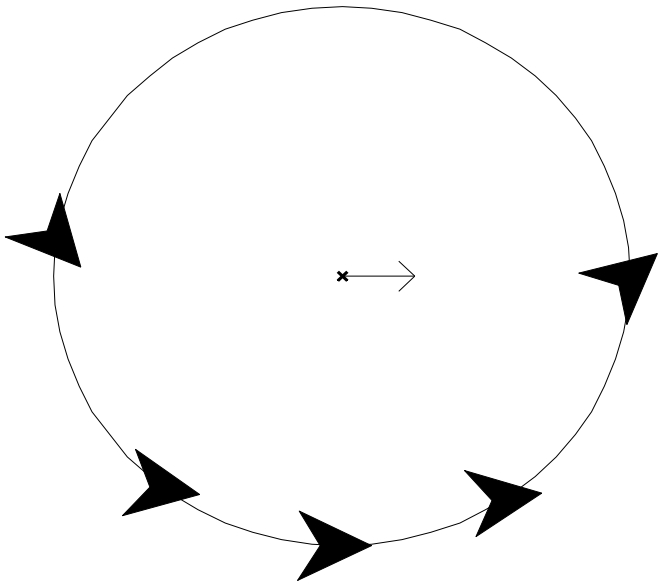


Figure 5.2: A target moving at 75% of UAV speed has a splay state configuration with 5 vehicles that corresponds to the spacing in this figure. Note that at the bottom of the orbit, the target and the UAV are moving in the same direction, so the UAV slowly turns the corner. However, at the top of the orbit, the UAV and the target are moving in opposite directions, so the UAV quickly moves around the arc.

constant airspeed, constant altitude UAV in wind, is given by

$$\begin{aligned} \dot{p}_N &= V_a \cos\psi + V_w \cos\psi_w \\ \dot{p}_E &= V_a \sin\psi + V_w \sin\psi_w \\ \dot{\psi} &= \tfrac{g}{V_a} \tan\phi \\ \dot{\phi} &= u \end{aligned} \tag{5.1}$$

where (p_N, p_E) are the (North, East) coordinates of the UAV in a flat earth model, ψ is the heading of the UAV (with the $\dot{\psi}$ equation given by the coordinated turn assumption), ϕ is the roll angle, V_a is the constant airspeed of the vehicle, V_w is the magnitude of the wind vector and ψ_w is the heading of the wind vector (note that this is not the meteorological definition of wind heading, i.e. ψ_w is the direction the wind is blowing *to* as opposed to the direction the wind is blowing *from*). In addition to these dynamics, a constraint on roll angle $-\phi_{\max} \leq \phi \leq \phi_{\max}$ is enforced so that stall conditions are avoided.

We consider the motion of the UAV relative to a target position. Let

$$\begin{aligned} x &= p_N - q_N \\ y &= p_E - q_E \end{aligned} \tag{5.2}$$

where $(q_N,\ q_E)$ is the position of the target. The dynamics of (5.1) become

$$\begin{aligned} \dot{x} &= V_a \cos\psi + W_x \\ \dot{y} &= V_a \sin\psi + W_y \\ \dot{\psi} &= \tfrac{g}{V_a} \tan\phi \\ \dot{\phi} &= u \end{aligned} \tag{5.3}$$

where $W_x = V_w \cos\psi_w - \dot{q}_N$ and $W_y = V_w \sin\psi_w - \dot{q}_E$. Target velocity and wind are indistinguishable with respect to the relative motion of the UAV to the target. This allows the control design to maintain constant airspeed and account for wind disturbances and target motion with only regard to $(W_x,\ W_y)$.

Model (5.3) can be reduced further by letting

$$u = \frac{g V_a \dot{\omega}}{g^2 + V_a^2 \omega^2}$$

where ω is the heading rate of the UAV, i.e. $\omega = \frac{g}{V_a} \tan\phi$. Model (5.3) then becomes the kinematic unicycle model

$$\begin{aligned} \dot{x} = &\ V_a \cos\psi + W_x \\ \dot{y} = &\ V_a \sin\psi + W_y \\ \dot{\psi} = &\ \omega \end{aligned} \tag{5.4}$$

where we constrain $|\omega| \leq \frac{g}{V_a} \tan(\phi_{\max})$ to ensure that $|\phi| \leq \phi_{\max}$. The constraint on ω can be thought of as a curvature constraint on the system kinematics from which it follows that the UAV can be considered a Dubins-type vehicle. This model 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 [62, 66, 67].

The heading design and analysis is performed at a level of abstraction greater than the unicycle level by computing a desired heading ψ^d and using it as a feed-forward term to the model (5.4). Feedback is then introduced at the control signal ω while maintaining the saturation constraints on ω. Let

$$\omega = \dot{\psi}^d + \nu \tag{5.5}$$

where ν is the feedback term driving ψ to ψ^d. This chapter shows that ψ^d can be chosen so that a team of UAVs with individual dynamics

$$\begin{aligned} \dot{x} &= V_a \cos\psi^d \\ \dot{y} &= V_a \sin\psi^d \end{aligned} \tag{5.6}$$

can reach the splay state configuration. Control gains in the calculation of ψ^d can then be chosen to allow the saturation constraints on ω to be satisfied. Note that ψ^d can be considered a sliding surface along which the specifications of the mission are satisfied. If ψ reaches ψ^d in finite time via the feedback term ν, then the overall system can be guaranteed to converge to the splay state configuration. Theoretically, a sliding mode controller of the form

$$\nu = \beta \text{sign}(\psi - \psi^d)$$

ensures that ψ reaches ψ^d in finite time, however in practice, a control law of the form

$$\nu = \beta \text{sat}\left(\frac{\psi - \psi^d}{\epsilon}\right)$$

is used, where β is a positive control gain. We do not show the overall system stability with this control strategy, but refer the reader to [68] where this choice of ν is shown to ensure path convergence for an arbitrary path in the single UAV case.

5.2.2 Orbit Dynamics

We will be concerned with the behavior of UAV teams while orbiting a target at a fixed radius R_{nom}. To analyze the stability of the orbit system, we make a change of variables by letting

$$\begin{aligned} R &= \sqrt{x^2+y^2} \\ \theta &= \tan^{-1}\left(\tfrac{y}{x}\right) \end{aligned} \tag{5.7}$$

where R is the distance of the UAV from the target and θ is the "clock angle" of the UAV around the orbit.

In the static target, no wind case (i.e. $W_x = W_y = 0$), the dynamics of R and θ can be calculated as follows. Let

$$\chi \triangleq \psi - \psi^p \tag{5.8}$$

be the difference between the actual heading, ψ, and the heading of the tangent vector to the orbit, i.e. $\psi^p = \theta + \pi/2$. Therefore $\dot{R}$ can be calculated as

$$\begin{aligned} \dot{R} &= \tfrac{d}{dt}\sqrt{x^2+y^2} \\ &= \tfrac{x\dot{x}+y\dot{y}}{\sqrt{x^2+y^2}} \\ &= \tfrac{V_a}{R}\left[x\cos\psi + y\sin\psi\right] \quad . \end{aligned}$$

Since $\psi = \chi + \theta + \pi/2$, we obtain

$$\dot{R} = \frac{V_a}{R}\left[-x\sin(\chi+\theta) + y\cos(\chi+\theta)\right] \quad .$$

Using the relations $\frac{x}{R} = \cos\theta$ and $\frac{y}{R} = \sin\theta$ we get that

$$\begin{aligned} \dot{R} &= -V_a\left[\cos\theta\sin(\chi+\theta) - \sin\theta\cos(\chi+\theta)\right] \\ &= -V_a\left\{\sin\chi\cos^2\theta + \cos\chi\sin\theta\cos\theta - \cos\chi\sin\theta\cos\theta + \sin\chi\sin^2\theta\right\} \\ \Rightarrow\ \dot{R} &= -V_a\sin\chi \quad . \end{aligned}$$

Similar arguments are used to derive the equation of motion for θ resulting in

$$\begin{aligned} \dot{R} &= -V_a \sin\chi \\ \dot{\theta} &= \frac{V_a}{R}\cos\chi \ . \end{aligned} \tag{5.9}$$

In the case of a moving target and/or wind, the motion is abstracted by calculating the path heading ψ^p, i.e. the heading which the UAV should be traveling if directly on the path. By accounting for target motion and wind via the ψ^p term, the radial orbit dynamics remain identical to those in (5.9) [68]. We show in Section 5.5 the calculation of ψ^p for moving targets.

To accommodate the multiple UAV splay state configuration, a spacing term is defined. For the static target, no wind scenario, the separation of the i^{th} agent from the angular mean of its neighbors is

$$\delta\theta_i = \frac{1}{2}\left((\theta_i - \theta_{i-1}) - (\theta_{i+1} - \theta_i)\right) \tag{5.10}$$

where a ring topology is assumed (i.e. addition is defined modulo N). The term $\delta\theta_i$ captures how far away agent i is from being equally spaced between its two immediate neighbors on the ring. When all agents are on the nominal radius with spacing terms $\delta\theta_i$ equal to zero, then the team has achieved the splay state configuration. Although the calculation of $\delta\theta_i$ is more complicated in the moving target case, the principle is the same: $\delta\theta_i$ captures how far away from the splay state configuration agent i is with regards to its immediate neighbors along the ring.

A visual representation of the notation used to describe the desired heading calculation is shown in Figure 5.3 where d_i is the radial error from the nominal radius, i.e. $d_i \triangleq R_i - R_{\text{nom}}$.

5.3 Heading Calculation for Non-Moving Targets

This section details the construction of a desired heading to achieve the splay state configuration in the case of zero wind and a non-moving target. The basis of the splay state configuration controller is the calculation of an appropriate heading

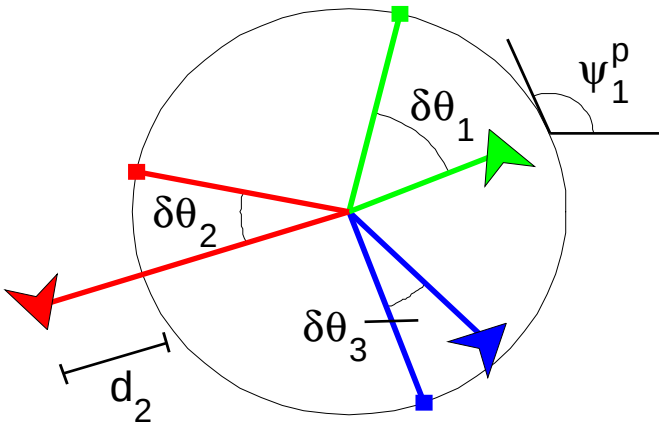


Figure 5.3: Spacing error and radial error are combined to construct a desired heading for each UAV. Radial error is determined by the distance from the desired orbit (d_i) and spacing error is the distance from the angular center of an agent's two immediate neighbors ($\delta\theta_i$).

command that steers the agents to the proper steady-state behavior. By creating a desired heading for the UAV, a reliable, robust heading controller can be used to track the heading commands. For a single UAV, a desired heading of the form

$$\psi^d = \psi^p + \tan^{-1}(kd) \tag{5.11}$$

will draw the agent onto the path, where d is the distance from the path and ψ^p is the heading along the path at $d = 0$ [68]. Using definition (5.8) equation (5.11) can be reduced to

$$\chi = \tan^{-1}(kd) \ . \tag{5.12}$$

Note that when d is large, the commanded heading is almost perpendicular to the heading along the path, effectively steering the UAV toward the path before beginning to follow it. For a simple orbit maneuver, ψ^p is selected to be tangent to the circle of interest along the ray connecting the agent and the target position. The radial distance of the agent from the nominal orbit constitutes d and a heading field constructed via (5.11) is shown in Figure 5.4. The gain k determines how aggressive the field is in steering the agent to the desired path.

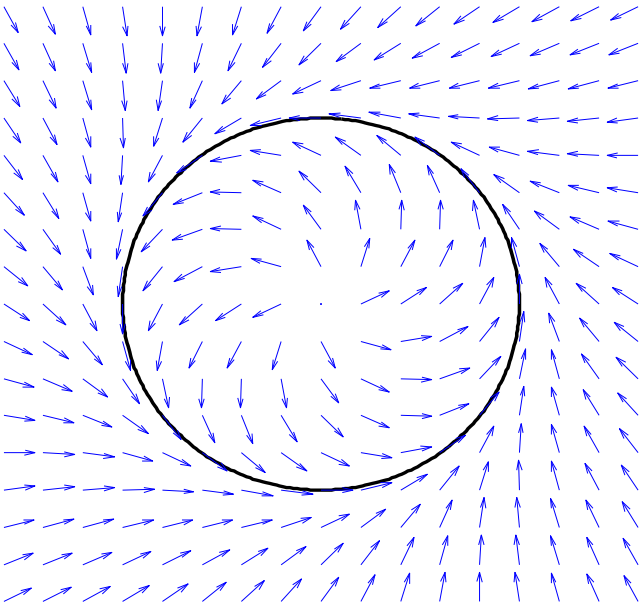


Figure 5.4: A single UAV orbiting a stationary target has a commanded heading computed at each point given by (5.11). Note that when the agent is far from the orbit, the heading steers it toward the target. As it gets near the desired trajectory, the desired heading transitions to tangent to the nominal circular motion.

The constraint on ω is satisfied when

$$\max|\omega| = \max|\dot{\psi}^d| + \beta \leq \omega_{\max}$$

where $\omega_{\max} = \frac{g}{V_a}\tan(\phi_{\max})$ and β is the maximum control allowed for the feedback control term (see Equation (5.5)). Due to the relationship in Equation (5.11), the term $\max|\dot{\psi}^d|$ can be bounded by

$$\max|\dot{\psi}^d| < \max|\dot{\psi}^p| + \max|\dot{\chi}| \ .$$

The term $\max|\dot{\psi}^p|$ can be determined using *a priori* knowledge or an estimate of the path to be tracked (e.g. moving orbit, straight line, etc.); for stationary orbits, $|\dot{\psi}^p| = V_a/R_{\text{nom}}$. The term $\max|\dot{\chi}|$ directly depends on the strength of the field

through the gain k. Recalling that $\chi = \tan^{-1}(kd)$ gives

$$|\dot{\chi}| = \left| \frac{k\dot{d}}{1+(kd)^2} \right| = \left| \frac{-kV_a \sin\chi}{1+(kd)^2} \right| \leq kV_a$$

which when coupled with knowledge of $\dot{\psi}^p$, the gain k can be chosen so as not to violate the UAV turn rate/roll angle constraints.

For a single UAV, a commanded heading of the form $\chi = \tan^{-1}(kd)$ guarantees asymptotic convergence to an orbit at radius R_{nom} about the target. A simple Lyapunov argument supports this assertion. Letting $W = \frac{1}{2}\chi^2$ and using (5.9) gives

$$\dot{W} = \chi\dot{\chi} = \frac{-kV_a\chi\sin\chi}{1+(kd)^2} \quad . \tag{5.13}$$

Since $\chi \in (-\pi/2, \pi/2)$ (χ is the output of an inverse tangent), the term $\chi\sin\chi$ is always greater than zero for nonzero χ. Therefore, $\dot{W} < 0$ and $\chi \to 0$ asymptotically. By LaSalle's invariance principle [69], it follows that $d \to 0$. Again we note that a complete proof for system (5.4) requires a sliding mode controller to guarantee that ψ reaches ψ^d in finite time, however, this can be relaxed as in [68]. Qualitatively, the commanded heading simply points the UAV directly toward the target if d is large and transitions to tangent to the orbit when near R_{nom}.

To account for spacing, the single agent heading command (5.11) is augmented as

$$\psi_i^d = \psi_i^p + \tan^{-1}(kd_i - \gamma\delta\theta_i) \tag{5.14}$$

where γ is a control gain weighting the value of spacing the UAVs to the value of converging to R_{nom}. The spacing term effectively increases the radius of the orbit when a UAV is too close to the agent in front of it and decreases the radius of the orbit if it is behind. This allows agents to "catch up" when the spacing is not at the desired state. An example of the heading field for an agent when $\delta\theta = \pi/2$ is shown in Figure 5.5. Notice the agent is drawn away from the nominal radius to allow the agent in front to increase its angular separation.

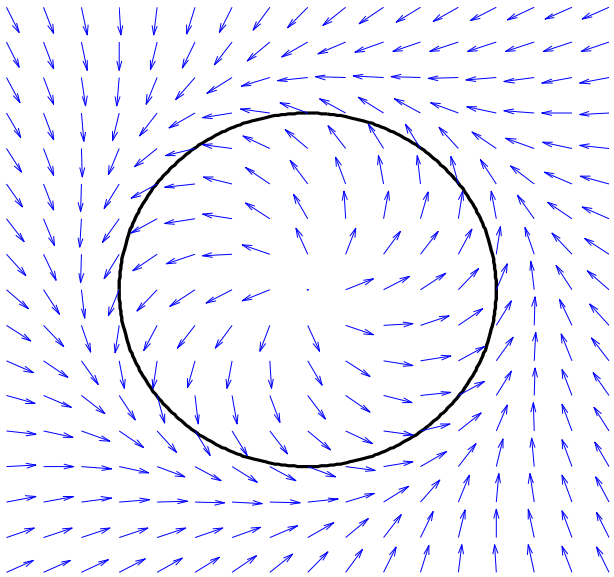


Figure 5.5: A single UAV orbiting a stationary target with spacing error $\pi/2$ has desired heading given by (5.14). Note that a positive spacing error will cause the agent to effectively increase its radius, allowing the neighbor in front to gain distance and increase their relative spacing.

By constructing $\delta\theta_i$ to be only a function of its immediate neighbors, the error signal (heading field calculation) is local to each agent in the system. This allows the implementation to be completely decentralized. The advantage to decentralization is that the overall system will scale to any number of agents and be robust to insertion and deletion of team members. When agents are tasked to leave the formation for high priority assignments, the rest of the group can adjust to a new configuration without any centralized planning. Similarly, if a new agent is added (e.g. returns from a high priority task) the group will adjust through local interaction without any global communication.

5.4 Stability Analysis

In the static target, no wind case, the splay state configuration coincides with the team members being equally spaced around an orbit. This section investigates the stability of the entire system when each agent follows the heading defined by (5.14). Figure 5.6 shows the behavior exhibited by a team of three UAVs.

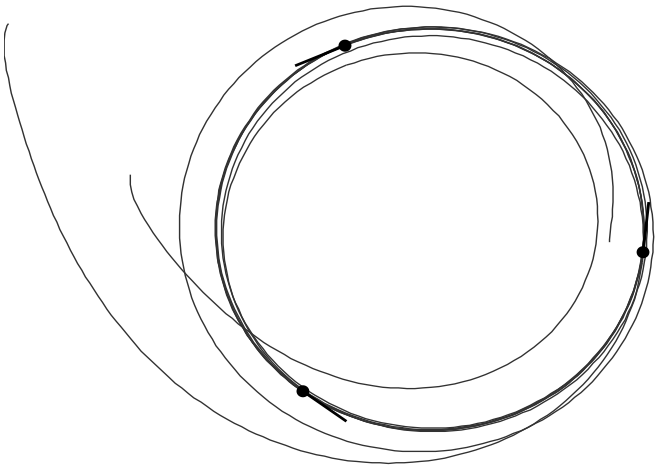


Figure 5.6: Three UAVs following the heading defined by (5.14) converge to the splay state configuration along a non-moving orbit.

A complete Lyapunov argument (or other method) may be used to determine the stability of the system to the splay state configuration. We have been unable to find a Lyapunov function that shows the stability of the entire system. For this reason, the convergence of the team of UAVs using (5.14) to the splay state configuration is argued as follows. We first show that the radial error is bounded by a function of the control gains k and γ. Near equilibrium, the overall system is shown to be exponentially stable. Finally, Monte-Carlo simulations are used to investigate system stability for initial conditions lying in the bounded region.

5.4.1 Ultimately Bounded

Lemma 7. *The system of agents described by (5.6) when following heading (5.14) is ultimately bounded in radial error d_i, i.e.*

$$|d_i| \leq R_\delta \tag{5.15}$$

where $R_\delta \triangleq \gamma\pi/k$ is less than R_{nom}.

Proof: For any agent, $\delta\theta_i$ is constrained to the region $(-\pi, \pi)$, i.e. agent cannot have an angular spacing error greater than π radians. If $|d_i| > R_\delta$, then

$$\begin{aligned} & \operatorname{sign}(k_i d_i - \gamma\delta\theta_i) = \operatorname{sign}(d_i) \\ \Rightarrow \quad & \operatorname{sign}(\chi_i) = \operatorname{sign}(d_i) \\ \Rightarrow \quad & \operatorname{sign}(\sin\chi_i) = \operatorname{sign}(d_i) \\ \Rightarrow \quad & \operatorname{sign}(-V_a \sin\chi_i) = \operatorname{sign}(-d_i) \\ \Rightarrow \quad & \operatorname{sign}(\dot{d}_i) = \operatorname{sign}(-d_i) \\ \Rightarrow \quad & d_i \dot{d}_i < 0 \ . \end{aligned}$$

Therefore, the Lyapunov function $W = d_i^2$ has a negative definite derivative whenever d_i is outside the bound (5.15). When $|d_i| > \gamma\pi/k$, the kd_i terms dominates the $\gamma\delta\theta_i$ term in (5.14) effectively steering the UAV to reduce radial error regardless of spacing error. Therefore, $|d|$ is decreasing when $|d| > \gamma\pi/k$ and so all d_i are ultimately bounded to the region $(-R_\delta, R_\delta)$. ■

5.4.2 Local Stability

The splay state configuration in the no wind, non-moving target case corresponds to all the UAVs traveling on the orbit equally spaced, i.e. $d_i = 0$ and $\delta\theta_i = 0$ for all agents on the team. The change of variables introduced in Section 5.2.2 allows analysis of the system dynamics where each UAV has equations of motion determined by (5.9). Rewriting (5.9) using the definition of $\delta\theta_i$ in (5.10) to evaluate the error signals for each agent, we obtain

$$\begin{aligned} \dot{d}_i &= -V_a \sin\chi_i \\ \dot{\delta\theta}_i &= \frac{V_a}{R_i}\cos\chi_i - \tfrac{1}{2}\left[\frac{V_a}{R_{i+1}}\cos\chi_{i+1} + \frac{V_a}{R_{i-1}}\cos\chi_{i-1}\right] \ . \end{aligned} \tag{5.16}$$

In the calculation of the linearization of (5.16), it is helpful to compute the partial derivatives of χ_i with respect to the system state variables d_i and $\delta\theta_i$. Since $\chi_i = \tan^{-1}(kd_i - \gamma\delta\theta_i)$, the partial derivatives evaluated at the equilibrium point

$d_i = 0$, $\delta\theta_i = 0$ are calculated as

$$\begin{array}{rcl} \frac{\partial \chi_i}{\partial d_i} & = & k \\ \frac{\partial \chi_i}{\partial d_{\neg i}} & = & 0 \\ \frac{\partial \chi_i}{\partial \delta\theta_i} & = & -\gamma \\ \frac{\partial \chi_i}{\partial \delta\theta_{\neg i}} & = & 0 \end{array} \tag{5.17}$$

where $\neg i$ represents any value in $\mathcal{I}$ not equal to i. The partial derivative of $\dot{d}_i$ can be calculated as

$$\frac{\partial}{\partial *}\left(\dot{d}_i\right) = \frac{\partial}{\partial *}\left(-V_a \sin\chi_i\right) = -V_a \cos\chi_i \left(\frac{\partial}{\partial *}\chi_i\right) \quad . \tag{5.18}$$

The matrix composing the partial derivatives of the system dynamics (5.16) has the structure

$$F = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \triangleq \left[\begin{array}{c|c} \frac{\partial}{\partial d_i}(\dot{d}_i) & \frac{\partial}{\partial \delta\theta_i}(\dot{d}_i) \\ \hline \frac{\partial}{\partial d_i}(\dot{\delta\theta}_i) & \frac{\partial}{\partial \delta\theta_i}(\dot{\delta\theta}_i) \end{array}\right] \quad . \tag{5.19}$$

Combining (5.18) with (5.17), the matrices A and B are calculated as $A = -kV_a I_N$ and $B = \gamma V_a I_N$ where I_N is the $N \times N$ identity matrix.

The linearization of the $\delta\theta$ dynamics reveals the ring structure inherent in the spacing calculation used to construct the desired heading. The function $\dot{\delta\theta}_i$ is composed of terms

$$\frac{V_a}{R_i}\cos\chi_i$$

which when linearized become

$$\frac{V_a}{R_i^2}\left(\frac{\partial}{\partial *}R_i\right)\cos\chi_i - \frac{V_a}{R_i}\sin\chi_i\left(\frac{\partial}{\partial *}\chi_i\right) \quad .$$

At the equilibrium, the only term that does not become zero is the term containing $\partial R_i/\partial d_i$. Note that since R_i does not depend on $\delta\theta_i$, the partial derivative with

respect to $\delta\theta_i$ will be zero. The linearized dynamics of $\delta\theta_i$ become

$$\begin{array}{rcl} \frac{\partial}{\partial d_i}\left(\dot{\delta\theta}_i\right) & = & \frac{-V_a}{{R_{\text{nom}}}^2} \\ \frac{\partial}{\partial d_{i\pm1}}\left(\dot{\delta\theta}_i\right) & = & \frac{1}{2}\frac{V_a}{{R_{\text{nom}}}^2} \\ \frac{\partial}{\partial \delta\theta_i}\left(\dot{\delta\theta}_{i,\neg i}\right) & = & 0\ . \end{array} \tag{5.20}$$

We conclude that the matrix D in (5.19) is simply the zero matrix of size $N \times N$ and matrix C is a circulant matrix

$$C = \frac{1}{2}\frac{V_a}{{R_{\text{nom}}}^2}\begin{bmatrix} -2 & 1 & 0 & \cdots & 0 & 1 \\ 1 & -2 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & 0 & \cdots & 1 & -2 \end{bmatrix}\ . \tag{5.21}$$

Of particular note is the structure of C

$$C = \frac{1}{2}\frac{V_a}{{R_{\text{nom}}}^2}\left(-2I_N + \mathcal{C}_N\right) \tag{5.22}$$

where

$$\mathcal{C}_N = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 1 \\ 1 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix} \tag{5.23}$$

is the adjacency matrix corresponding to the ring graph of size N. The eigenvalues of F can be formulated in terms of the eigenvalues of C which are known using results from algebraic graph theory [59].

Lemma 8. *Consider the matrix*

$$F = \begin{bmatrix} -kV_aI_N & \gamma V_aI_N \\ C & 0_N \end{bmatrix} \tag{5.24}$$

where C is given by (5.21), I_N is the $N \times N$ identity matrix and 0_N is an $N \times N$ matrix of zeros. The eigenvalues of F are given by

$$\lambda_j = -\frac{1}{2}kV_a \pm \sqrt{\left(\frac{1}{2}kV_a\right)^2 + \gamma V_a \mu_j} \quad \text{for } j = 1 \ldots N \tag{5.25}$$

where

$$\mu_j = \frac{1}{2}\frac{V_a}{{R_{\text{nom}}}^2}\left(2\cos\left(\frac{2\pi}{N}(j-1)\right) - 2\right) \tag{5.26}$$

is an eigenvalue of C.

Proof: We begin by showing that the eigenvalues of C are given by (5.26). From (5.22) we conclude that

$$\mu_j = \frac{1}{2}\frac{V_a}{{R_{\text{nom}}}^2}\left(-2 + \gamma_j\right)$$

where γ_j is an eigenvalue of $\mathcal{C}_N$. Results from algebraic graph theory show that the eigenvalues of $\mathcal{C}_N$ are

$$\gamma_j = 2\cos\left(\frac{2\pi}{N}(j-1)\right) \quad \text{for } j = 1 \ldots N \ .$$

Let λ be an eigenvalue of F and x its corresponding eigenvector. Partition x into blocks corresponding with the blocks of F, i.e. $x = \begin{bmatrix} x_d^T & x_{\delta\theta}^T \end{bmatrix}^T$ where both x_d and $x_{\delta\theta}$ are of length N. The eigenvector relationship $Fx = \lambda x$ can be written

$$-kV_a x_d + \gamma V_a x_{\delta\theta} = \lambda x_d \ \Rightarrow \ \gamma V_a x_{\delta\theta} = (\lambda + kV_a)\, x_d \tag{5.27}$$

and

$$Cx_d = \lambda x_{\delta\theta} \ . \tag{5.28}$$

From (5.27) we see that

$$x_{\delta\theta} = \frac{\lambda + kV_a}{\gamma V_a} x_d \tag{5.29}$$

which when applied to (5.28) yeilds

$$Cx_d = \left(\frac{\lambda(\lambda + kV_a)}{\gamma V_a}\right) x_d \ .$$

Note that this is exactly the eigenvector relationship for the matrix C where $Cx = \mu x$ for

$$\mu = \left(\frac{\lambda(\lambda + kV_a)}{\gamma V_a} \right) \quad .$$

Solving this for λ yields Equation (5.25). ■

Theorem 9. *Consider the matrix F as defined in (5.24). All eigenvalues except for $\lambda = 0$ of F are located in the open left half plane. Additionally, the eigenvectors associated with $\lambda = 0$ and $\lambda = -kV_a$ span a subspace of $\mathbb{R}^{2N}$ orthogonal to the remaining $2N - 2$ eigenvectors of F.*

Proof: Equation (5.25) gives the relationship of the eigenvalues of F to the eigenvalues of C. Only a single eigenvalue of C is equal to zero, all other $N - 1$ values are strictly less than zero. The zero eigenvalue in C maps to the eigenvalues $\lambda = -kV_a$ and $\lambda = 0$ in F. The remaining eigenvalues of C (all strictly less than zero) have discriminant strictly less than $(\frac{1}{2}kV_a)^2$ thus ensuring that each λ has real part in the open left half plane.

The proof of Lemma 8 gives the relationship between the eigenvectors of C and those of F via (5.29) where x_d is the eigenvector of C corresponding to eigenvalue

$$\mu = \left(\frac{\lambda(\lambda + kV_a)}{\gamma V_a} \right) \quad .$$

Since C is a symmetric matrix, its eigenvectors form an orthonormal basis of $\mathbb{R}^N$. Note that C has constant row sums of zero, so the eigenvector associated with the zero eigenvalue of C is the vector of all ones, $\mathbf{1}$. Due to the orthogonality of the eigenvectors of C, $\mathbf{1}^T u_j = 0$ for all eigenvectors of C, $u_j \neq \mathbf{1}$. Using (5.29), the eigenvectors for $\lambda = 0$ and $\lambda = -kV_a$ are

$$x_0 = \begin{bmatrix} \mathbf{1} \\ \frac{k}{\gamma}\mathbf{1} \end{bmatrix}, \quad x_{-kV_a} = \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \end{bmatrix} \quad . \tag{5.30}$$

The inner product of these eigenvectors with all other eigenvectors of F can be written as

$$\begin{bmatrix} \mathbf{1}^T & \frac{k}{\gamma}\mathbf{1}^T \end{bmatrix} \begin{bmatrix} u_j \\ \frac{\lambda+kV_a}{\gamma V_a}u_j \end{bmatrix} = 0 \quad \text{and} \quad \begin{bmatrix} \mathbf{1}^T & \mathbf{0}^T \end{bmatrix} \begin{bmatrix} u_j \\ \frac{\lambda+kV_a}{\gamma V_a}u_j \end{bmatrix} = 0 \ .$$

■

Corollary 10. *The linearization of system (5.16) is exponentially stable.*

Proof: Linearization of (5.16) yields the state equation $\dot{x} = Fx$ where F is given in equation (5.24), and whose solution is $x(t) = e^{Ft}x_0$. By Theorem 9 all but one eigenvalue is in the open left half plane, so any part of the initial condition x_0 that lies in the span of the eigenvectors associated with those eigenvalues exponentially decays to zero. By definition of $\delta\theta_i$, the constraint

$$\sum_{i=1}^{N} \delta\theta_i = 0 \tag{5.31}$$

must hold for any state vector associated with the original system. The eigenvectors associated with $\lambda = 0$ and $\lambda = -kV_a$ are given in (5.30). These eigenvectors form a subspace orthogonal to all other eigenvectors in the linearized system. To lie in the subspace spanned by the eigenvectors (5.30), all $\delta\theta_i$ must be equal. However, the only vector $\boldsymbol{\delta\theta}$ that satisfies the constraint (5.31) and is in this subspace is $\boldsymbol{\delta\theta} = \mathbf{0}$, which is either along the eigenvector associated with $\lambda = -kV_a$ or in the subspace spanned by the remaining eigenvectors of the system. In other words, it is impossible to have an initial condition in the subspace spanned by the eigenvector associated with $\lambda = 0$. Therefore, the initial condition x_0 lies in the space spanned by eigenvectors whose eigenvalues are in the open left half plane and the linearized system is exponentially stable. ■

5.4.3 Global Stability

The system (5.16) is ultimately bounded to $d_i \in (-R_\delta,\ R_\delta)$, $\delta\theta_i \in (-\pi,\ \pi)$ and locally asymptotically stable. Monte-Carlo simulations are used to infer the stability

of the system in the remaining region between the ultimate bound and the equilibrium path.

The Monte-Carlo simulations use the model (5.4) with desired heading given by (5.14). For team sizes $N = 2$, 3, 4, 5, and 6, a set of 10,000 simulations with random initial conditions in d_i and $\delta\theta_i$ were run to verify the stability of the system. An error metric

$$e(t) = \sqrt{\sum_{i=1}^{N} d_i(t)^2 + \delta\theta_i(t)^2}$$

captures the error from the splay state configuration at time t. The largest error at $t = 100$ seconds over all 50,000 simulations was $2e^{-4}$ indicating that the actual region of convergence is likely to be global.

5.5 Extension to Moving Targets

The ability for a UAV to orbit a target in the presence of wind or target motion is crucial. Modifications to the static target, no wind case can be made to allow UAVs to track moving targets.

To extend the approach of (5.14) to moving targets, the path heading term ψ^p must be calculated to allow a UAV to remain on a moving orbit. Essentially, the steady-state behavior of a UAV on the orbit is determined by ψ^p: while following ψ^p at $d = 0$, a UAV should remain on the moving orbit.

Consider the behavior of a particle orbiting a constant speed target at fixed radius R_n then

$$\begin{aligned} x^p(t) &= R_n \cos(\theta(t)) + W_x t \\ y^p(t) &= R_n \sin(\theta(t)) + W_y t \end{aligned} \tag{5.32}$$

where W_x and W_y are the velocity of the orbit center. Differentiating (5.32) results in the expression

$$\begin{aligned} \dot{x}^p &= -R_n \dot{\theta} \sin\theta + W_x \\ \dot{y}^p &= R_n \dot{\theta} \cos\theta + W_y \ . \end{aligned} \tag{5.33}$$

The path heading is chosen as

$$\psi^p = \tan^{-1}\left(\frac{\dot{y}^p}{\dot{x}^p}\right) \tag{5.34}$$

which is the direction of the vector that is tangent to the moving orbit. To ensure that the UAV maintains constant airspeed, the magnitude of the tangent vector must equal V. This constraint allows the calculation of $\dot{\theta}$ from (5.33) as

$$\begin{aligned} V_a^2 = (\dot{x}^p)^2 + (\dot{y}^p)^2 &= \left(-R_n\dot{\theta}\sin\theta + W_x\right)^2 + \left(R_n\dot{\theta}\cos\theta + W_y\right)^2 \\ \Rightarrow \dot{\theta}^2\left(R_n^2\right) + \dot{\theta}\left(2R_nW_y\cos\theta - 2R_nW_x\sin\theta\right) &+ \left(W_x^2 + W_y^2 - V_a^2\right) = 0 \end{aligned}$$

$$\begin{aligned} \Rightarrow \dot{\theta} = -\tfrac{1}{R_n}\left(W_y\cos\theta - W_x\sin\theta\right) \pm \\ \tfrac{1}{R_n}\sqrt{\left(W_y\cos\theta - W_x\sin\theta\right)^2 - \left(W_x^2 + W_y^2 - V_a^2\right)}\ . \end{aligned} \tag{5.35}$$

The discriminant in (5.35) shows that when the magnitude of the velocity of the target is greater than the speed of the UAV, a real solution does not exist. In practical terms, this means that for the agent to properly maintain its orbit around the target, the speed of the wind plus the speed of the target cannot exceed the speed of the UAV.

The turn rate constraint of the UAV must also be accounted for in determining the allowable magnitude of motion that can be feasibly tracked. Disregarding the other components of heading rate,

$$\left|\dot{\psi}^p\right| \leq \frac{g}{V_a}\tan(\phi_{\max}) \tag{5.36}$$

ensures that the path satisfies the turn rate constraints. The maximum value of $\dot{\psi}^p$ depends on V_w, the magnitude of the motion in the system (note $V_w^2 = W_x^2 + W_y^2$). To ensure that the orbit can feasibly be followed with regard to the turn constraints of the UAV, V_w must satisfy

$$\frac{(V_w + V_a)^2}{R_nV_a} \leq \frac{g}{V_a}\tan(\phi_{\max})\ . \tag{5.37}$$

Intuitively, a UAV can follow a moving target in wind if the magnitude of the wind and target velocity are not too great to violate the velocity or turn rate constraints of the UAV. For example, a UAV with maximum bank angle of 35 degrees, airspeed of 15 meters per second and desired orbit of 100 meters can track a target with speed less than 11.2 m/s.

With ψ^p determined by (5.34), a desired heading of (5.11) can be used for a single UAV to follow a moving target in the presence of wind given that the turn rate constraint of the UAV is satisfied. For multiple UAVs, the definition of the splay state configuration is used to develop a spacing error term. Note that achieving equal angle spacing around a moving orbit is impossible when the velocity of the UAVs is held constant. For this reason, the actual time along the steady-state orbit between neighbors is used to compute the error from the splay state configuration. Similar to the static target case, the timing error is computed by assuming that all UAVs are on the desired orbit (i.e. $d_i = 0$). Consider two agents on the orbit with clock angles θ_i and θ_j. The time difference from agent i to agent j is given by $T_{i\to j} = t - t_0$ such that $\theta(t) = \theta_j$ where $\theta(t)$ is determined by solving the initial value problem

$$\begin{aligned} \dot{\theta} &= -\tfrac{1}{R_n}\left(W_y\cos\theta - W_x\sin\theta\right) \pm \\ &\qquad \tfrac{1}{R_n}\sqrt{\left(W_y\cos\theta - W_x\sin\theta\right)^2 - \left(W_x^2 + W_y^2 - V_a^2\right)} \qquad (5.38) \\ \theta(t_0) &= \theta_i \ . \end{aligned}$$

The timing error for a specific agent i can then be defined as

$$\delta t_i = \frac{1}{2}\left(T_{(i-1)\to i} - T_{i\to(i+1)}\right) \ . \qquad (5.39)$$

The δt term is used in exactly the same manner as the $\delta\theta$ term in the static target case, i.e. a desired heading is calculated as

$$\psi_i^d = \psi_i^p + \tan^{-1}(kd_i - \gamma\delta t_i) \ . \qquad (5.40)$$

Many of the stability notions from the non-moving target case carry over to the moving target case. A maximum δt exists since agents can only be of finite angle apart. Therefore, for large errors in radial distance d, the kd_i term will dominate the heading calculation and force the system to be ultimately bounded. A linearization of the system dynamics for the moving target case also shows many similarities to static case. In particular the upper two blocks of the state matrix are identical to the blocks in the static target linearization. We postulate that the lower blocks are identical up to a positive scale factor, i.e. the circulant structure of the lower left block is preserved which allows us to conclude linear stability via the same arguments as in the static target case. Additionally, Monte-Carlo simulations are used to indicate that the system converges to the splay state configuration in the moving target case. For team sizes $N = 2$, 3, and 4, a set of 1,000 simulations with random initial conditions in d_i, δt_i and V_w were run to verify the stability of the system. An error metric

$$e(t) = \sqrt{\sum_{i=1}^{N} \delta t_i(t)^2}$$

captures the error from the splay state configuration at time t. The largest error at $t = 100$ seconds over 3,000 simulations was 0.5 indicating that control (5.40) leads to convergence to the splay state configuration. Figure 5.7 shows typical behavior of 4 UAVs orbiting a moving target. The timing error from the splay state configuration for this scenario is shown in Figure 5.8.

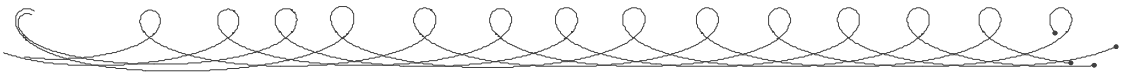


Figure 5.7: Trajectories of 4 UAVs orbiting a moving target trace out routes similar to those in this figure.

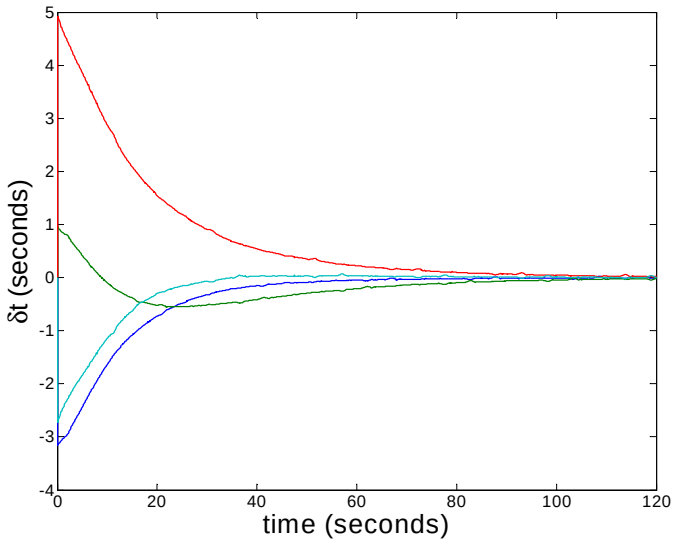


Figure 5.8: Error from the splay state configuration for 4 UAVs tracking a moving target is driven to zero using (5.40).

5.6 Simulation and Hardware Results

The splay state controller is based upon choosing a heading that draws the UAVs to the splay state configuration. The design of the heading command is accomplished by assuming a simple kinematic model given by (5.4). To validate the design, the splay state controller is tested in high-fidelity simulation. Each UAV is simulated with full 6 degree-of-freedom dynamics model with aerodynamic parameters that match the small UAVs flown at BYU [70]. Additionally, the human interface and autopilot code are emulated to match actual flight conditions as closely as possible.

Trajectories of three UAVs that loiter at fixed locations and are then commanded to reach the splay state configuration are shown in Figure 5.9. The radial error of the agents is approximately one meter and the spacing error about three degrees. These errors are due mainly to the update rate of the team — each UAV only communicates to its neighbors when a new GPS packet is received, at approximately 1 Hertz.

Actual flight tests of the algorithm are also used to validate the algorithm design and its performance. A team of small UAVs is assembled and programmed to reach the splay state configuration for a fixed orbit around a stationary point. Each

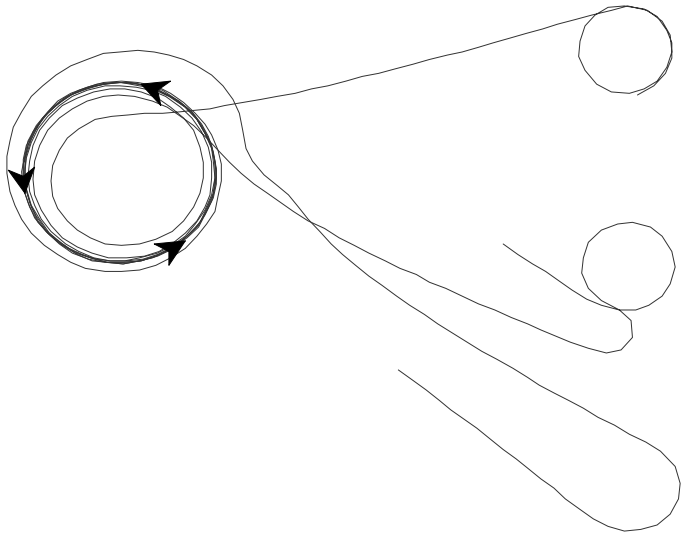


Figure 5.9: High fidelity simulation results of the splay state controller indicate that the method can be effective in actual implementation.

UAV is equipped with an autopilot and navigation sensors to follow the desired heading determined by the cooperative splay state controller. The team communicates via wireless modem to other team members and to a ground station where a human operator monitors the status and behavior of the team.

A primary difficulty when flight testing small UAVs in groups is the use of the available communication bandwidth. In our architecture, all UAVs share the same channel and are polled successively by the ground station software to ensure that packets do not collide causing interference for the entire team. The disadvantage to this architecture is that the ground station must request and then wait for a response before polling the next team member. When packets are dropped due to interference or loss of line-of-sight, agents can remain without communication for long periods of time. During experiments, UAVs were frequently out of communication for periods up to five seconds. The infrequent communication naturally degrades the performance of the algorithm since each agent relies on knowledge of the states of its neighbors.

Despite complications arising from bandwidth constraints, hardware results indicate that the algorithm can still reach the splay state configuration as can be seen in Figure 5.10. Telemetry packets transmitted from the two UAVs were collected

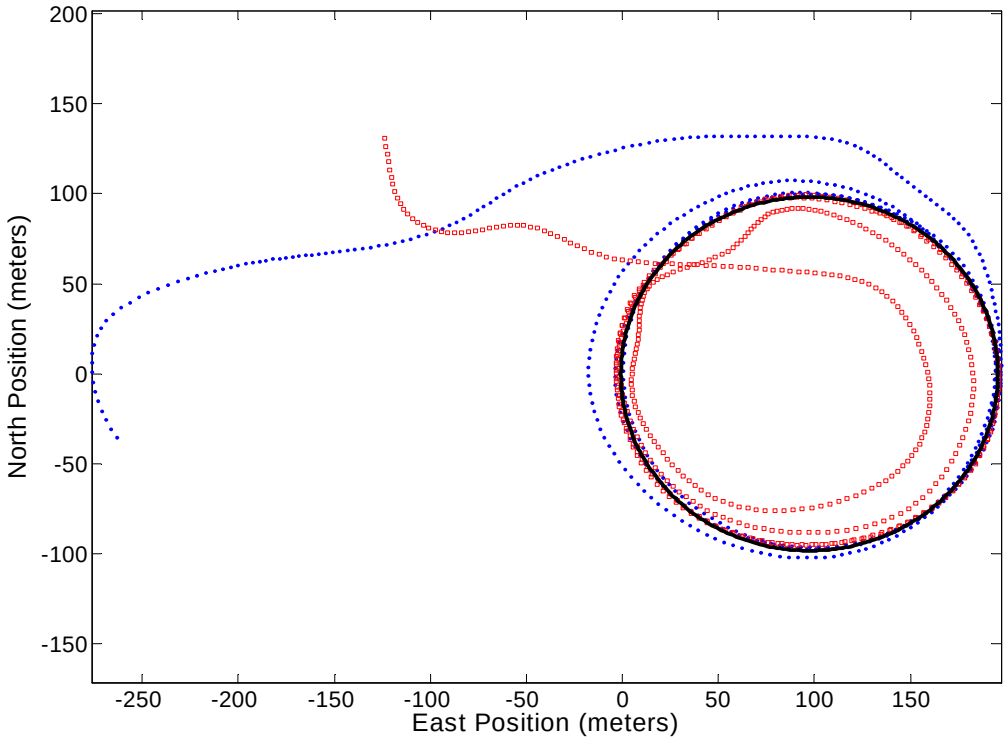


Figure 5.10: Flight test results for two UAVs using the cooperative splay state controller. Notice that one UAV (blue dots) orbits a larger radius while the second (red squares) orbits a smaller radius so that their relative spacing converges to the splay state configuration.

and gaps in the data filled by spline interpolation. Using this data, it was verified that the radial error was on the order of 4 meters and the error from the splay state configuration about 5 degrees. Considering the coarse update rate, wind conditions, differences between airframes and sensor inaccuracies, we consider these very positive results validating the splay state controller.

5.7 Conclusions and Future Work

This chapter has developed a decentralized splay state controller for a team of UAVs monitoring a target. In the static case (i.e. non-moving target and no wind), the controller spaces UAVs equally around an orbit centered on the target. The decentralized nature of the control strategy allows the the team to be robust to insertion, deletion and re-assignment of team members. The controller is shown

to be linearly stable in the static target case and Monte-Carlo simulations indicate global stability in all cases. By defining an appropriate measure of spacing around the orbit, the splay state configuration can be reached for moving targets in the presence of wind. High fidelity simulation results and flight test experiments show that the controller is practical and robust.

There are still many open questions in regards to the convergence of a team of UAVs to the splay state configuration. Monte-Carlo simulations indicate that the region between the ultimate bound and the equilibrium is stable, but a formal proof of this assertion remains an open problem. Additionally, the design of the commanded heading is based on a low-order UAV model. Extending the analysis to the model (5.1) and finding an appropriate control u, rather than relying on a sliding mode inner-loop control, is also an important extension.

Chapter 6

Perimeter Surveillance

6.1 Introduction

Perimeter surveillance algorithms form the basis for effective monitoring in a number of applications including monitoring oil spills [71], contaminant clouds [72], algae bloom [73], forest fires [2, 74], and border security [75, 3]. The literature in this area can roughly be decomposed into two main groups: sensor technology used for perimeter detection; and algorithms used to gather data along the perimeter effectively.

Sensors that have been investigated for small fixed border scenarios, such as warehouse surveillance, include cameras [76], ultrasound [77], and radar [78]. In Ref. [76], the authors discuss algorithms that use image data from multiple cameras to determine a perimeter breach. Peralta [77] uses a chain of ultrasound sensors with a simple detection scheme to identify border crossings. Research has also been done using existing airport radar equipment to identify when people or vehicles come too close to runways [78]. For spill monitoring and other dynamic perimeter scenarios, surveillance vehicles are equipped with chemical concentration sensors [1], infrared cameras [74], or standard optical cameras [71].

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. These advantages create unique requirements for the cooperation algorithms that control teams of small UAVs. 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 the cooperation strategies. Finally, the computational burden should stay 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 an algorithm.

We are particularly interested in monitoring borders that are of unknown shape and size and possibly changing in time such as would be encountered in a forest fire or chemical spill monitoring scenario. 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 circumstances such as changing perimeters (spill monitoring, forest fire surveillance) or very large perimeters (border patrol).

A number of researchers have investigated similar problems of monitoring and tracking changing perimeters with autonomous vehicles. The MDARS project [79] is a joint effort between the 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 [3]. The entire team of vehicles communicates to a central location where sensor data is fused and waypoint commands are issued [75]. 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 by the team to a base location where the state of the perimeter is displayed and human operators make decisions.

Teams of unmanned water vehicles have been proposed as a way to track algae bloom and oil spills. Bertozzie et al. in Ref. [80] present an algorithm for monitoring a perimeter with multiple agents when each 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 form a complete picture of 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 [81].

Clark and Fierro propose a similar method for oil spill perimeter tracking using multiple vehicles [71]. 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. [80], neither the efficiency nor the convergence of the algorithms are shown analytically. In addition, neither explicitly address the problem of limited communication range.

Susca, Martinez and Bullo address the issue of approximating a changing border with a set of interpolation points [4]. 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 chapter, 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 accident or refueling.

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.

The perimeter surveillance problem is posed in Sections 6.2 and 6.3. Section 6.4 presents our solution using a coordination variable [34] approach and compares it to both averaging and centralized solutions. The method is extended to changing perimeters in Section 6.5 and to account for constrained UAV turning radius in Section 6.6. Simulation and hardware results are presented in Section 6.7 and 6.8. Finally, Section 6.9 gives our conclusions.

6.2 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 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 such as 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 an autopilot similar to the one described in Ref. [70]. The autopilot maintains constant altitude and each UAV on the team is given a unique altitude assignment. The autopilot has been tuned so that the closed-loop system exhibits a first-order response to roll and

airspeed commands. Under these assumptions, the kinematic equations of motion for a single UAV can be written as

$$\begin{aligned}
\dot{p}_N &= V\cos\psi + w_N \\
\dot{p}_E &= V\sin\psi + w_E \\
\dot{\psi} &= \frac{g}{V}\tan\phi \\
\dot{V} &= \alpha_V(V^c - V) \\
\dot{\phi} &= \alpha_\phi(\phi^c - \phi),
\end{aligned} \tag{6.1}$$

where $\mathbf{p} = (p_N, p_E)^T$ is the inertial position of the UAV, ψ, ϕ, and V are the heading, roll angle, and airspeed, g is the gravitational constant, $\mathbf{w} = (w_N, w_E)^T$ is the windspeed, and V^c and ϕ^c are the airspeed and roll angle commands given to the autopilot. The first order response of the autopilot to airspeed and roll angle commands are quantified by the parameters α_V and α_ϕ. In addition to these kinematics, a constraint on roll angle $-\phi_{\max} \leq \phi \leq \phi_{\max}$ is enforced to ensure the safety of the UAV. The presence of wind and the roll angle constraint impair the maneuverability of the agents.

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 6.3. Section 6.4 then introduces and proves the convergence of an algorithm for reaching the desired behavior while accounting for most of the complications.

6.3 Linear Perimeter Surveillance

We reduce the general perimeter surveillance problem of Section 6.2 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 6.1). Corresponding UAV agents follow their reference points along the perimeter as described in Section 6.6. 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.

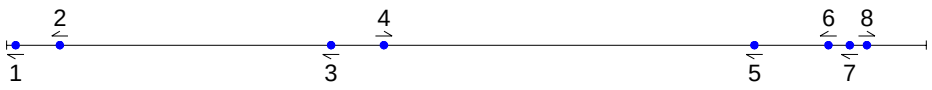


Figure 6.1: Example scenario where 8 agents monitor a linear perimeter.

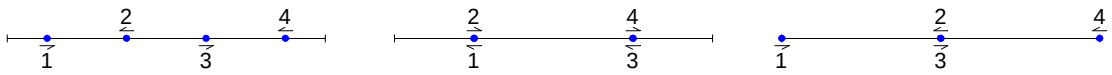
The performance of a particular monitoring algorithm can be measured by the latency associated with information about points along the perimeter. Let P

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 of V and data transmission only occurs when the agent is 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 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 6.2. The agents are uniformly distributed along the perimeter (Figure 6.2(a)) and each agent meets its neighbors at the end of its segments (Figures 6.2(b) and 6.2(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 6.2 it can be seen that information gathered at neighbor meeting locations travels to all other locations along the



(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.

Figure 6.2: Information exchange pattern that allows information about the state of the perimeter to be available at any point along the perimeter.

perimeter in the shortest time possible. This can be seen by noting that after two agents meet and gather information about the perimeter at their meeting place, each will take this information at speed V to any other place 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 at the neighbor meeting locations is carried to all other points along the perimeter at the highest possible speed.

Definition 2 (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 $1 \ldots N$. Consider two sets of team agent locations on the perimeter:*

1. *Agent $i \in 1 \ldots N$ is located at $\lfloor i + \frac{1}{2}(-1)^i \rfloor P/N$*
2. *Agent $i \in 1 \ldots N$ 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 2 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 6.2(c)). The next set of agents starts by pairing agent 1 and 2 at position $x = P/N$ and spacing the remaining pairs at $2P/N$ intervals (see Figure 6.2(b)). Note that for each agent i, the pair of positions in Definition 2 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 6.4. In addition to converging to the low-latency exchange configuration, the algorithm will address deletion and insertion of team members and variable length perimeters.

6.4 Decentralized Solution

This section derives a decentralized algorithm to reach the low-latency exchange configuration defined in Definition 2. One way to approach such a problem is to determine the *coordination variables* [28] or minimum amount of information necessary to achieve cooperation. For this problem, 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 the coordination variables. For each agent $i \in 1 \ldots N$, let

$$\xi_i = \begin{pmatrix} P_{R_i} \\ P_{L_i} \\ N_{R_i} \\ N_{L_i} \end{pmatrix}$$

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 $n = N_{R_i} + 1$. 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 $i \in 1 \ldots N$, $P_{R_i} + P_{L_i}$ matches the true perimeter length and $N_{R_i} + N_{L_i} + 1$ matches the actual number of agents on the team.

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. The following algorithm 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.

Algorithm 1: Neighbor Escort

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 $s = \lfloor n \pm \frac{1}{2}(-1)^n \rfloor P/N$.
    Calculate shared border position $p = s_i \cap s_j$.
    Travel with neighbor j to shared border p.
    Set direction to monitor own segment.
else if *reached perimeter endpoint* **then**
    Reverse direction.
else
    Continue in current direction.

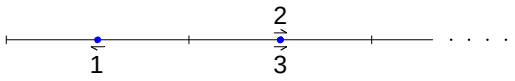
For every consecutive pair of agents, there is a single position where their segments border each other. When each agent has a knowledge of the length of the perimeter and its order in the team, then the endpoints of its responsible segment are computed. The endpoint shared with a neighbor is the shared border position to which both will travel together in the first phase of Algorithm 1. 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 11. *Let the perimeter length P and number of agents N be fixed. If all agents have correct coordination values, then Algorithm 1 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 1 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 6.3. 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 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 only found on their respective segments. This implies that the low-latency exchange configuration of Section 6.3 has been reached.

(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.

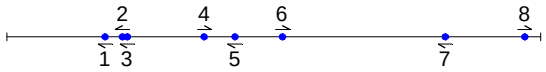
Figure 6.3: Possible cases for rendezvous of agent 1 with its neighbor.

The worst case situation occurs when all agents are stacked infinitesimally close at one end of the perimeter and are traveling toward the other. Once T has passed all agents are at the opposite end of the perimeter where they meet both 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 $2T$. ■

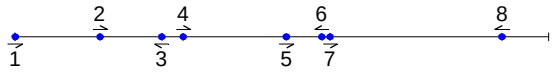
Figure 6.4 shows two simple scenarios with 8 agents spreading out over a fixed perimeter where each agent begins with correct coordination variables. The positions of agents along the perimeter is indicated vertically with the time axis shown horizontally. The lattice structure indicates that the desired steady-state behavior has been reached since agents turn around at precisely their desired neighbor rendezvous locations. Note that the agents require very few meetings with each other to converge to the proper configuration.

6.4.1 Comparison with Centralized Algorithm

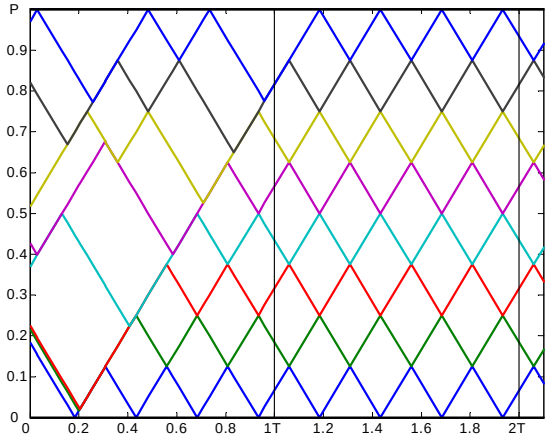
To understand the characteristics of Algorithm 1, it is useful to compare its performance with other methods of perimeter surveillance. A centralized method for reaching the low-latency exchange configuration is to compare the initial positions of the team with all possible team locations in the low-latency exchange configuration and find the one that requires the shortest convergence time.



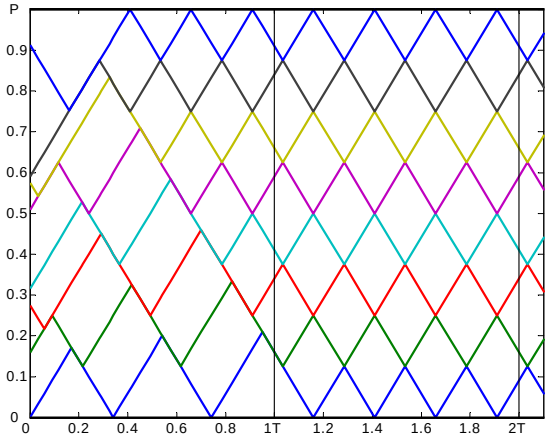
(a) Initial positions and directions for a group of 8 agents in scenario A.



(b) Initial positions and directions for a group of 8 agents in scenario B.



(c) Position of agents along the perimeter over time for scenario A.



(d) Position of agents along the perimeter over time for scenario B.

Figure 6.4: Team behavior in two scenarios for point agents whose behavior is governed by Algorithm 1. 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.

Let $\mathcal{Q}$ be the set of team positions during the desired steady-state operation where an element $q \in \mathcal{Q}$ consists of N positions, q_i corresponding to the position of agent i in the low-latency exchange configuration. Note that the set of all team configurations that satisfy the low-latency exchange configuration can be parameterized by the position of the first agent

$$q_i = (i-1)\frac{P}{N} + \begin{cases} \left(\frac{P}{N} - q_1\right) & \text{if } i \text{ is even} \\ q_1 & \text{otherwise .} \end{cases} \tag{6.2}$$

Therefore, if the position of the first agent is known in the low latency configuration, then for a perimeter of length P, q_1 is on the interval $[0, \frac{P}{N}]$ and all other positions can be calculated using Equation (6.2). The centralized method is to command the

team located at p_i, $i = 1 \ldots N$ to converge to q^* where

$$q^* = \arg\min_{q \in \mathcal{Q}} \max_{i=1\ldots N} |p_i - q_i| \quad . \tag{6.3}$$

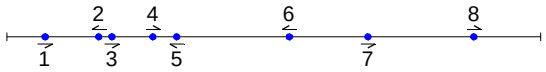
In other words, the optimal solution is to pick the low-latency team configuration that is closest to the current position of the team. During the transition from the initial position to the nearest low-latency configuration position q^*, agents reach their correct position and loiter there until the remaining team members have reached their respective positions.

In the worst case scenario where all agents are located at one end of the perimeter, the centralized algorithm converges in time T, twice as fast as the decentralized method. Figure 6.5 shows a comparison of the centralized algorithm and Algorithm 1. Note that the centralized algorithm requires agents to wait or loiter at the proper location until all agents have reached q^*. This is indicated by the straight lines in Figure 6.5.

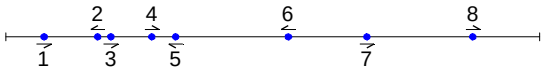
Monte-Carlo simulations indicate that the centralized algorithm reaches the low-latency exchange configuration on average $0.67T$ seconds faster than the decentralized method (standard deviation of $0.17T$ seconds). The maximum time difference between the centralized algorithm and Algorithm 1 was $0.998T$ seconds corresponding to the theoretical worst case difference. The centralized algorithm requires complete knowledge of the state of the team and explicit cooperation of all team members. The value of Algorithm 1 is that its performance is comparable to the optimal solution in speed, but is implemented in a decentralized, robust way.

6.4.2 Comparison with Consensus Method

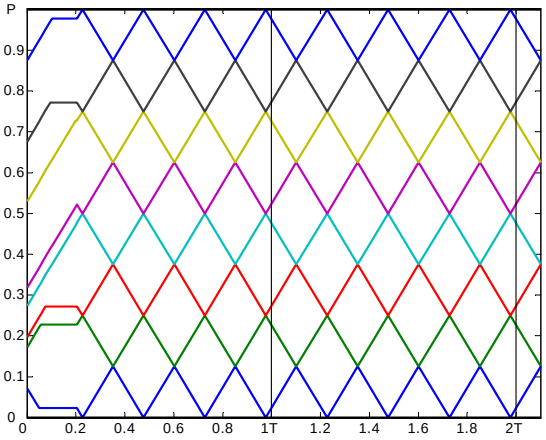
The second method to which we compare Algorithm 1 is a distributed consensus algorithm modified for perimeter surveillance. The standard consensus problem for a group of agents is to ensure that as time progresses each agent approaches a consistent understanding of their shared information. For example, one method of coming into consensus is for each agent to repeatedly average its associated variable



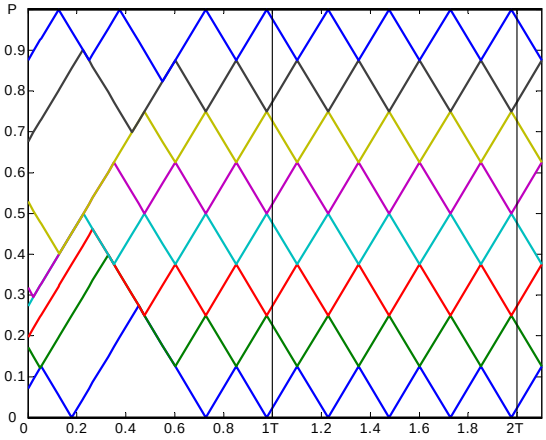
(a) Initial positions and directions for a group of 8 agents.



(b) Initial positions and directions for a group of 8 agents.



(c) Position of agents along the perimeter over time when using the centralized algorithm (6.3).



(d) Position of agents along the perimeter over time when using Algorithm 1.

Figure 6.5: Team behavior in a comparison of Algorithm 1 and the centralized algorithm (6.3). 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. Straight lines indicate that an agent is maintaining its current position along the perimeter. Note that the centralized algorithm requires agents to wait for the rest of the team to settle into the optimal starting configuration while Algorithm 1 reaches the low-latency exchange configuration after some interaction time.

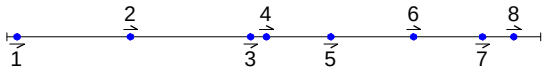
with those communicated from its immediate neighbors. If the interaction graph among the team contains a spanning tree, then the coordination variable of each agent will asymptotically approach a constant shared value and the team is said to asymptotically reach consensus [82].

Adapting a consensus method to the perimeter surveillance problem involves defining the value associated for each agent and a strategy for updating those values. Let the length of the segment for which an agent is responsible be the value associated with that agent. Consider the rendezvous of two agents on the perimeter. When agents meet, they communicate the length of their respective segments and average to find the midpoint of their shared segment. Both travel together to the midpoint

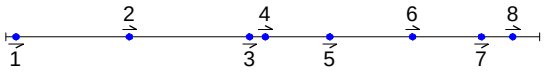
of their shared segment [83] and seperate with an updated value for how much of the perimeter each is responsible for. For every pair of agents, their shared segment is defined by endpoints determined by the locations where each agent met its other neighbor.

The difficulty with this method is that the value to which the team will converge *must* be P/N where P is the length of the perimeter; otherwise, agents would be continuously overlapping or neglecting part of the perimeter. A specialization of the general consensus problem to the average consensus problem can be made which ensures that the team will converge to the exact average of the initial values. The only remaining difficulty is initializing the system so that the segment lengths associated with the team of agents sum to P. We do this by assuming that agents are launched with a value of zero with the exception of the first agent who travels to the end of the perimeter and initializes its value to P. This approach has three consequences. First, although the algorithm can account for arbitrary perimeter length, the perimeter must remain fixed. Second, loss of an agent during the mission will remove its segment length from the knowledge of the team. In each case, the prerequisites for average consensus would be violated and the team would fail to converge to the true value of P/N. Finally, convergence is, in general, asymptotic in nature rather than in finite time as Algorithm 1 guarantees. Figure 6.6 shows the performance of the average consensus algorithm compared to Algorithm 1. In addition to the above limitations of fixed perimeter length and asymptotic convergence, the consensus method seems to exhibit poor transient response.

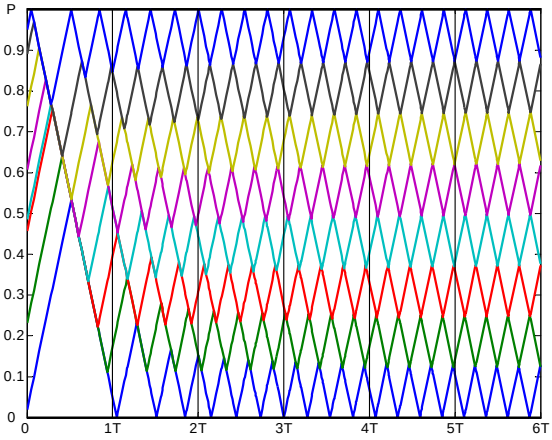
Algorithm 1 relies only on interactions of an agent with its immediate neighbors on the perimeter and yet it converges to the low-latency exchange configuration in finite time. In Section 6.5 we show that the decentralized nature of the algorithm allows the team to accommodate loss or reassignment of agents. In the event of a perimeter breach, an agent can be assigned to follow the intruder while the rest of the team reconfigures to monitor the border in its absence. Since the algorithm converges in finite time, the loss in perimeter coverage is quickly compensated. This



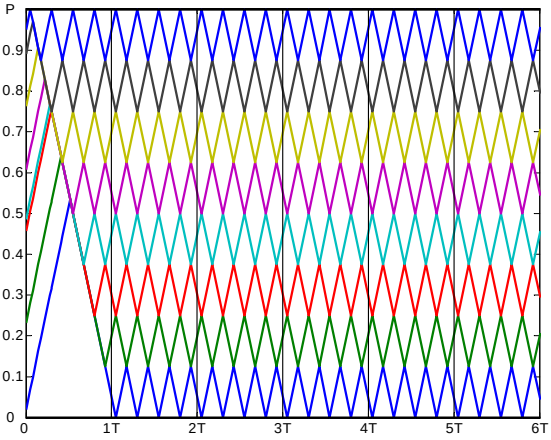
(a) Initial positions and directions for a group of 8 agents.



(b) Initial positions and directions for a group of 8 agents.



(c) Position of agents along the perimeter over time when using the centralized algorithm (6.3).



(d) Position of agents along the perimeter over time when using Algorithm 1.

Figure 6.6: Team behavior in a comparison of Algorithm 1 and a consensus method (6.3). 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. Note that the consensus method converges asymptotically while Algorithm 1 reaches the low-latency exchange configuration in finite time.

same natural reconfiguration behavior is desirable in the event of refueling and agent loss due to hazardous conditions.

6.5 Changing Perimeters

Theorem 11 ensures finite time convergence to the low-latency exchange configuration of Section 6.3 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 1 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 11 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 2: Variable Neighbor Escort

if *agent i (left) rendezvous with neighbor j (right)* **then**
  Update perimeter length and team size:
    $P_{R_i} = P_{R_j}$
    $P_{L_j} = P_{L_i}$
    $N_{R_i} = N_{R_j} + 1$
    $N_{L_j} = N_{L_i} + 1$.
  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 $s = \lfloor n \pm \frac{1}{2}(-1)^n \rfloor P/N$.
  Calculate shared border position $p = s_i \cap s_j$.
  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.

Algorithm 2 operates in the same manner as Algorithm 1, 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 pro-

gresses 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} from the communication from agent 1. At this point, the coordination variables are correct and Theorem 11 ensures that the low-latency exchange configuration will be reached in finite time.

Theorem 12. *Let the perimeter length P and number of agents N be fixed. Algorithm 2 ensures that the low-latency exchange configuration is achieved in finite 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 2. 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.

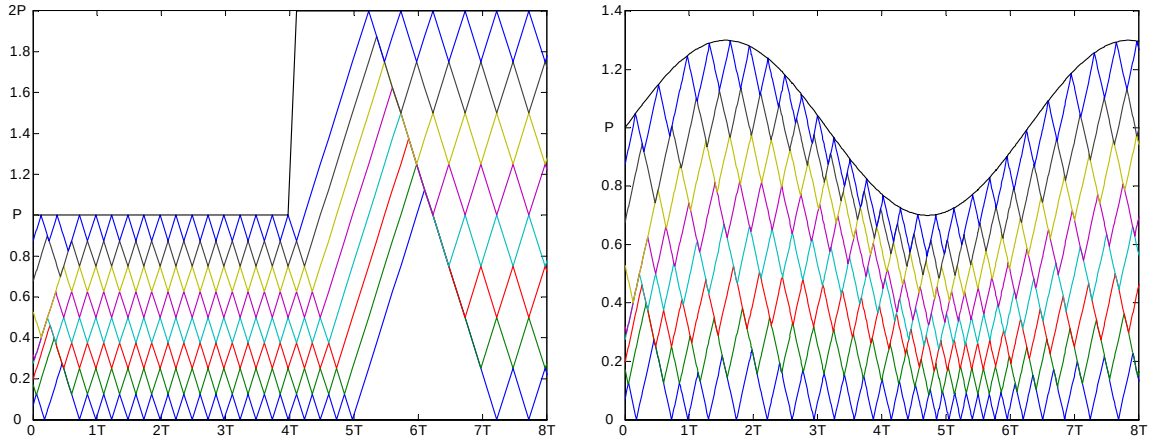
Order the N agents from the left edge of the perimeter as $1, \ldots, N$ and consider the actions of agent 1. Agent 1 is guaranteed to visit the left endpoint of the perimeter either after an escort from agent 2 or immediately due to initial conditions. Once agent 1 has visited the perimeter endpoint, both N_{L_1} and P_{L_1} are correct due to the section of Algorithm 2 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 11 ensures that the low-latency exchange configuration will be met in finite time. ■

Algorithm 2 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 come 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 2 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 6.7 shows agents tracking a perimeter with a step change in size and a perimeter with sinusoidal growth. The algorithm 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.

Algorithm 1 can also be extended to account for long communication events. In a perimeter imaging scenario, agents survey the perimeter segment for which they are responsible and when each meets its neighbor it must transmit large amounts of data. In this case, agent meetings cannot be instantaneous, rather a fixed amount of time is allotted for agents to loiter at the meeting location to allow longer commu-



(a) Positions of agents over time for step change in perimeter length.

(b) Positions of agents over time for sinusoidal perimeter growth.

Figure 6.7: Team behavior of agents tracking changing perimeters using Algorithm 2 to continuously update the coordination variables. Agents learn the size of the perimeter and number of agents on the team through repeated interaction with other team members.

nication events. After an agent finishes escorting its neighbor, both loiter together for a pre-determined amount of time. Figure 6.8 shows a scenario involving long communication events.

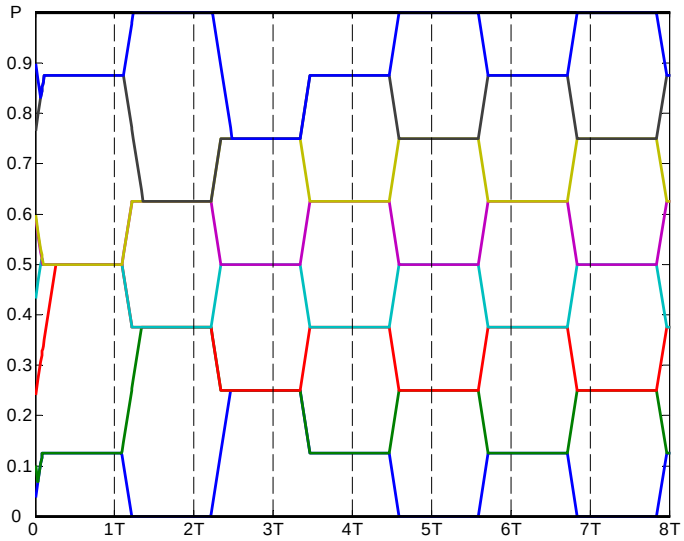


Figure 6.8: Example scenario where Algorithm 1 is modified to account for long rendezvous timing.

6.6 UAV Agents

Algorithm 1 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, it cannot precisely follow a reference point that can instantaneously turn around. The purpose of this section is to investigate the application of Algorithm 1 when the dynamics of the UAVs are considered.

In Section 6.3 agents are modeled as points that could communicate only when touching. Now consider UAVs flying at constant velocity with nominal turning radius R. A maneuver for reversing direction with constrained turning radius is shown in Figure 6.9 where the UAV follows arcs along minimum turn radius circles to complete the path direction reversal.

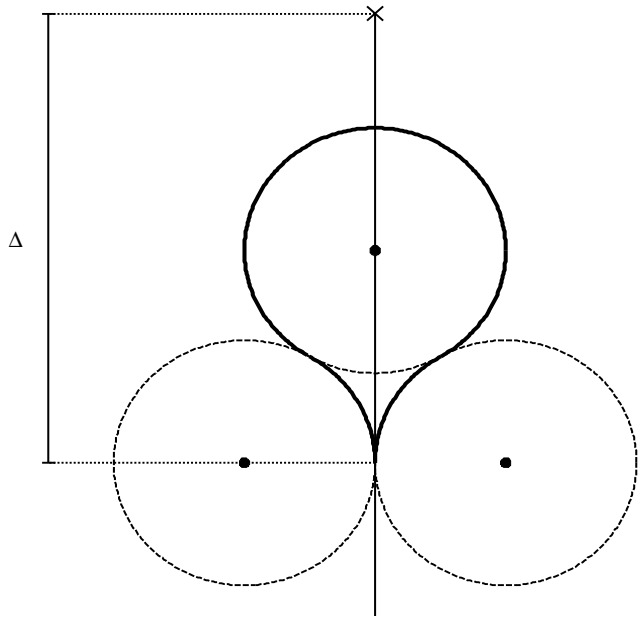


Figure 6.9: U-turn maneuver that satisfies the constrained turning radius of the UAV.

The distance required to travel around the U-turn trajectory in Figure 6.9 is $\Delta = \frac{7}{3}\pi R$ where R is the nominal turning radius of the UAV. To allow the reference point to follow the pattern dictated by Algorithm 1, 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. Since each requires a distance of $\frac{7}{3}\pi R$ to turn around, the minimum communication radius allowed must be $\frac{14}{3}\pi R$ so that both can be aware of an imminent rendezvous.

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 similar to the behavior of the agents that have long communication events. In other words, when UAVs meet, they loiter the rendezvous point for a specified amount of time before continuing with the algorithm and data gathering.

A method for reducing the amount of turning around by the team 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.

6.7 Simulation Results

To verify the feasibility of implementing Algorithm 1 on a team of UAVs, a high fidelity simulation is performed. Each UAV is simulated with full 6 degree-of-freedom dynamics model with aerodynamic parameters that match the small UAVs flown at BYU [70]. 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 [70] 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 6.10 shows the simulation results by plotting the normalized position of each UAV along the length of the 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 6.10 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.

6.8 Flight Test Results

The decentralized cooperative-surveillance algorithm was further validated by hardware flight tests using the experimental testbed described in Ref. [70]. Figure 6.11 displays the normalized position of two UAVs along the perimeter while Figure 6.12 shows the inertial position plots that were generated from the actual telemetry files

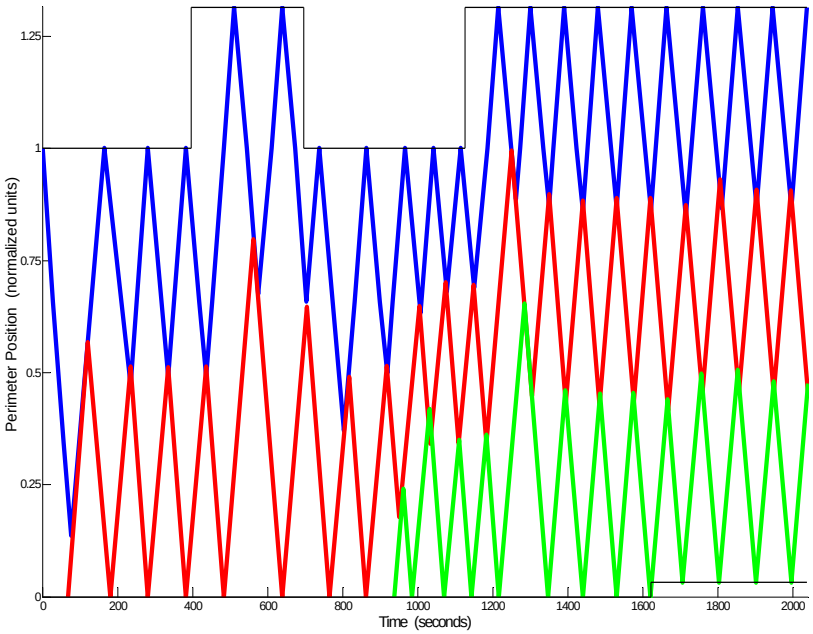


Figure 6.10: Simulation results showing the normalized position of each UAS along the perimeter. Changes to the perimeter length occurred at approximately 400, 700, 1100, and 1600 seconds. The third agent was introduced at approximately 950 seconds. The sharp peaks are a result of the coordination variables being reset.

of the UAVs. Figure 6.12 demonstrates the algorithm 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.

The algorithm 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 agent (red) traveled to the shared border before turning around. At this point the agents had reached the steady-state configuration. As seen in Figure 6.11, there was 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

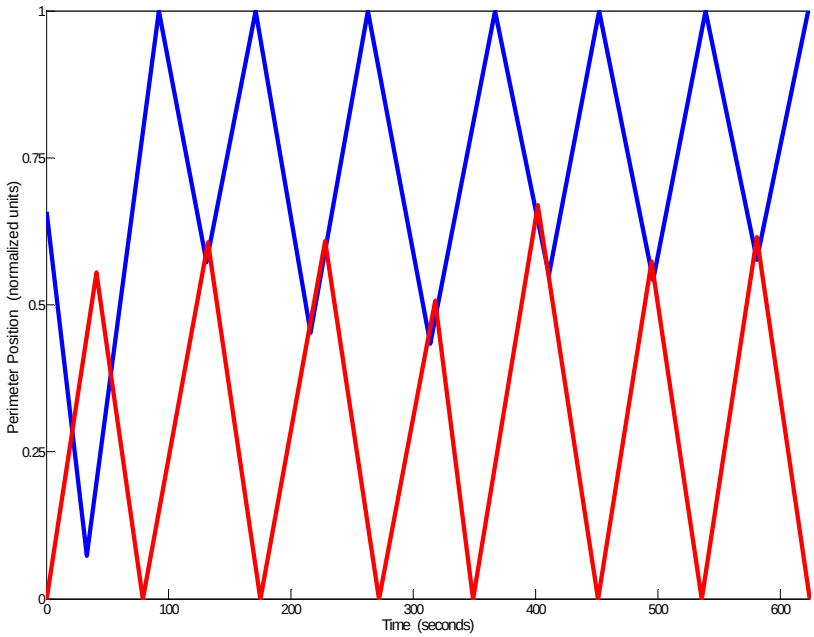
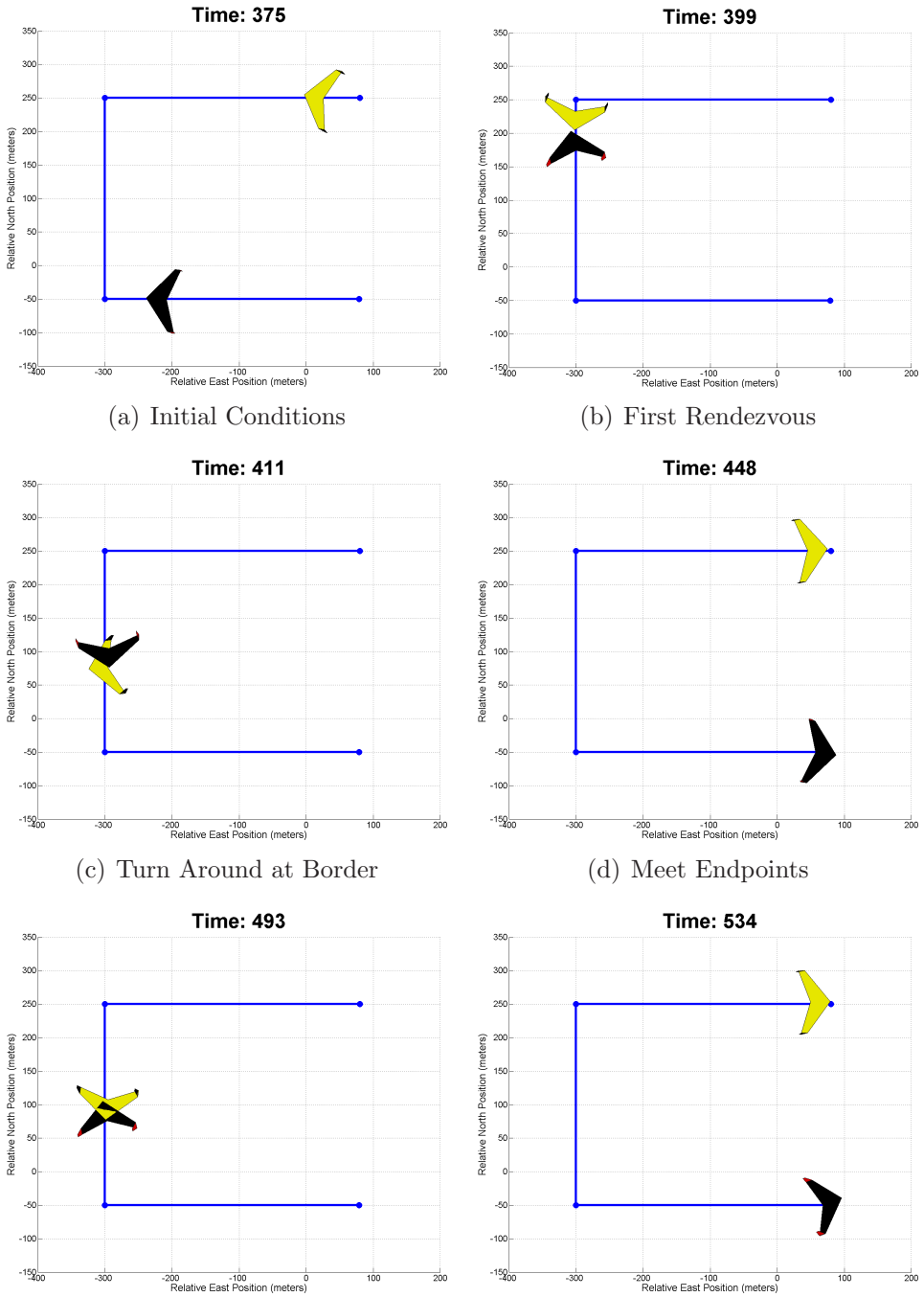


Figure 6.11: Experimental results showing the normalized position of each UAS along the perimeter. The decentralized cooperative-surveillance algorithm was started at approximately 50 seconds.

disturbance of the wind, the agents were still able to effectively distribute themselves evenly along the perimeter.

6.9 Conclusions

This chapter 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 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. Simulation and flight tests were performed to validate the effectiveness of the algorithm.



(a) Initial Conditions (b) First Rendezvous

(c) Turn Around at Border (d) Meet Endpoints

(e) Second Rendezvous (f) Meet Endpoints Again

Figure 6.12: Various plots generated from the actual telemetry files of the UASs collected during the experimental flight tests. These demonstrated the functionality of the distributed spread algorithm, 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.

Chapter 7

Conclusions and Future Work

There is no unified framework into which all (or even most) cooperative control problems fall. For this reason, different cooperative control algorithms are difficult to compare in a reasonable manner. The natural ability of an algorithm to deal with loss of agents and disrupted communication is invaluable to the practical deployment of an algorithm on a UAV team. We presented two algorithms that are provably complete and yet retain robustness to loss of agents.

For single target surveillance scenarios, a cooperative splay state controller was developed. The splay state controller requires only immediate neighbor communication and is completely decentralized and robust. Hardware tests validate its applicability to real UAV systems.

Perimeter surveillance was investigated and an algorithm to coordinate a team of UAVs designed. The robust perimeter surveillance algorithm requires very little communication and is optimal in steady-state and near optimal in the transition region. Additionally, the algorithm can be proven to converge in finite-time and shows practicability through flight tests.

In contrast to these algorithms we showed a centralized algorithm for cooperative target prosecution. This algorithm requires centralization due to the tight coupling of the tasks and the desired efficiency. Unfortunately, the computational complexity scales exponentially with the size of the team, and so can be burdensome to implement.

One possible method to convert centralized algorithms to decentralized ones are consensus methods. We prove that most consensus algorithms are input-to-state stable and therefore can be put in cascade with a centralized algorithm to achieve

decentralization in some cases. Additionally, we addressed the average-consensus problem and postulated that when extra bandwidth is available, ad hoc networks may yield better performance.

The key elements of cooperative control are objective coupling, level of communication, completeness, robustness, and efficiency. For loosely coupled problems, robust decentralized algorithms can be designed to satisfy all of these elements and are practical cooperative control solutions.

7.1 Future Work

There are a number of directions for future research for each of the topics covered. The tradeoffs between ad hoc networking and consensus schemes should be more deeply investigated to determine how each scales with team size and at what point the performance of one is provably better.

Proof of convergence of the splay state controller of Chapter 5 or reformulation to allow further analytical results is warranted in the single target surveillance scenario. Currently, only local stability can be shown analytically with Monte Carlo simulation used to imply global stability.

Future work in perimeter surveillance is in evaluating the performance of Algorithm 2 under changing perimeter conditions. Specifically, a metric for perimeter coverage could be developed and used to show the performance of the algorithm in a worst-case changing perimeter situation. This could be formulated in a game-theoretic framework, pitting the perimeter growth against the coverage of algorithm evaluated by the developed metric. Further development of the perimeter surveillance algorithm is also needed to extend the surveillance algorithm to accommodate circular and networked perimeters. The effect of wind on the team behavior during the operation is also a topic of interest.

Bibliography

[1] D. Marthaler and A. L. Bertozzi, "Tracking environmental level sets with autonomous vehicles," in *Recent Developments in Cooperative Control and Optimization*. Kluwer Academic Publishers, 2004. 1, 83

[2] 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. 1, 83

[3] H. R. Everett, "Robotic security systems," *IEEE Instrumentation & Measurement Magazine*, vol. 6, no. 4, pp. 30–34, Dec. 2003. 1, 83, 84

[4] S. Susca, S. Martinez, and F. Bullo, "Monitoring environmental boundaries with a robotic sensor network," *IEEE Transactions on Control Systems Technology*, (accepted for publication). 1, 85

[5] V. V. Prabhu, "Stable fault adaptation in distributed control of heterarchical manufacturing job shops," *IEEE Transactions on Robotics and Automation*, Feb. 2003. 1

[6] P. K. C. Wang, J. Yee, C. Y. Xia, M. Mokuno, and F. Y. Hadaegh, "Cooperative control of a magnetically levitated inferometer: Experimental study," *IEEE Transactions on Control Systems Technology*, July 2005. 1

[7] B. M. Braun, G. P. Starr, J. E. Wood, and R. Lumia, "A framework for implementing cooperative motion on industrial controllers," *IEEE Transactions on Robotics and Automation*, June 2004. 1

[8] K. Yamaji, M. Sato, K. Kato, M. Goto, and T. Kawai, "Cooperative control between large capacity hvdc system and thermal power plant," *IEEE Transactions on Power Systems*, May 1999. 1

[9] P. Ögren, E. Fiorelli, and N. E. Leonard, "Cooperative control of mobile sensor networks: Adaptive gradient climbing in a distributed environment," *IEEE Transactions on Automatic Control*, Aug. 2004. 1

[10] D. B. Kingston and C. J. Schumacher, "Time-dependent cooperative assignment," in *Proceedings of the American Control Conference*, 2005. 1, 2, 21

[11] D. H. A. Maithripala and S. Jayasuriya, "Radar deception through phantom track generation," in *Proceedings of the American Control Conference*, 2005. 1

[12] P. Varaiya, "Smart cars on smart roads: Problems of control," *IEEE Transactions on Automatic Control*, vol. 38, pp. 195–207, Feb. 1993. 1

[13] V. Gazi and K. M. Passino, "Stability analysis of swarms," *IEEE Transactions on Automatic Control*, vol. 48, no. 4, pp. 692–697, Apr. 2003. 2

[14] D. G. Luenberger, *Linear and Nonlinear Programming.* Addison-Wesley, 1984. 5

[15] C. Schumacher, P. Chandler, and S. Rasmussen, "Task allocation for wide area search munitions via iterative network flow," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2002. 5

[16] C. Schumacher, P. Chandler, S. Rasmussen, and D. Walker, "Task allocation for wide area search munitions with variable path length," in *Proceedings of the American Control Conference*, 2003. 5

[17] C. Schumacher, P. Chandler, M. Pachter, and L. Pachter, "UAV task assigment with timing constraints," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2003. 5, 8

[18] ——, "UAV task assignment with timing constraints via mixed-integer linear programming," in *Proceedings of the AIAA 3rd Unmanned Unlimited Systems Conference*, 2004. 5

[19] S. J. Rasmussen, J. W. Mitchell, A. G. Sparks, T. Shima, and P. R. Chandler, "Use of state-space search to improve the performance of assignment algorithms for autonomous UAVs," in *Proceedings of the IEEE Conference on Decision and Control*, 2004. 5

[20] D. Turra, L. Pollini, and M. Innocenti, "Real-time UAVs task allocation with moving targets," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2004. 5

[21] M. Alighanbari, Y. Kuwata, and J. P. How, "Coordination and control of multiple UAVs with timing constraints and loitering," in *Proceedings of the American Control Conference*, 2003. 5, 6

[22] J. Cortes, S. Martinez, and F. Bullo, "Coordinated deployment of mobile sensing networks with limited-range interactions," in *Proceedings of the IEEE Conference on Decision and Control*, 2004. 6

[23] E. Frazzoli and F. Bullo, "Decentralized algorithms for vehicle routing in a stochastic time-varying environment," in *Proceedings of the IEEE Conference on Decision and Control*, 2004. 6

[24] C. Schumacher, P. Chandler, S. Rasmussen, and D. Walker, "Path elongation for UAV task assignment," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2003. 6

[25] D. Bertsimas and S. S. Patterson, "The traffic flow management rerouting problem in air traffic control: A dynamic network flow approach," *Transportation Science*, vol. 34, pp. 239–255, 2000. 6

[26] A. Robertson, G. Inalhan, and J. P. How, "Formation control strategies for a separated spacecraft interferometer," in *Proceedings of the American Control Conference*, June 1999. 21

[27] R. W. Beard, J. Lawton, and F. Y. Hadaegh, "A coordination architecture for formation control," *IEEE Transactions on Control Systems Technology*, vol. 9, no. 6, pp. 777–790, Nov. 2001. 21

[28] T. W. McLain and R. W. Beard, "Coordination variables, coordination functions, and cooperative timing missions," in *Proceedings of the American Control Conference*, 2003. 21, 92

[29] N. A. Lynch, *Distributed Algorithms*. Morgan Kaufmann Publishers, Inc., 1996. 21, 49

[30] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," in *Advances in Ultra-Dependable Distributed Systems*, N. Suri, C. J. Walter, and M. M. Hugue, Eds. IEEE Computer Society Press, 1995. 21

[31] L. Lamport, "The part-time parliament," *ACM Transactions on Computer Systems*, vol. 16, pp. 133–169, May 1998. 21

[32] A. Jadbabaie, J. Lin, and A. S. Morse, "Coordination of groups of mobile autonomous agents using nearest neighbor rules," *IEEE Transactions on Automatic Control*, vol. 48, no. 6, pp. 988–1001, June 2003. 22, 27, 35

[33] W. Ren and R. W. Beard, "Consensus of information under dynamically changing interaction topologies," in *Proceedings of the American Control Conference*, 2004. 22, 35, 38

[34] W. Ren, R. W. Beard, and T. W. McLain, "Coordination variables and consensus building in multiple vehicle systems," in *Proceedings of the Block Island Workshop on Cooperative Control*. Springer-Verlag Series: Lecture Notes in Control and Information Sciences, 2004. 22, 27, 86

[35] L. Moreau, "Stability of multiagent systems with time-dependent communication links," in *IEEE Transactions on Automatic Control*, 2005. 22, 35, 38

[36] Z. Lin, M. Broucke, and B. Francis, "Local control strategies for groups of mobile autonomous agents," in *Proceedings of the IEEE Conference on Decision and Control*, Dec. 2003, pp. 1006–1011. 22

[37] ——, "Local control strategies for groups of mobile autonomous agents," *IEEE Transactions on Automatic Control*, pp. 622–629, 2004. 22

[38] L. Moreau, "Leaderless coordination via bidirectional and unidirectional time-dependent communication," in *Proceedings of the IEEE Conference on Decision and Control*, Dec. 2003, pp. 3070–3075. 22

[39] D. P. Spanos, R. Olfati-Saber, and R. M. Murray, "Approximate distributed Kalman filtering in sensor networks with quantifiable performance," in *Proceedings of the International Conference on Information Processing in Sensor Networks*, 2005. 22, 35, 51

[40] R. Olfati-Saber, "Distibuted Kalman filter with embedded consensus filters," in *Proceedings of the IEEE Conference on Decision and Control*, 2005. 22, 35

[41] R. Olfati-Saber and R. M. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, Sept. 2004. 22, 35, 41, 42, 54

[42] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," in *IFAC World Congress*, Barcelona, Spain, 2002. 22

[43] ——, "Graph Laplacians and stabilization of vehicle formations," in *IFAC World Congress*, Barcelona, Spain, 2002. 22

[44] R. Olfati-Saber and R. M. Murray, "Consensus protocols for networks of dynamic agents," in *Proceedings of the American Control Conference*, June 2003, pp. 951–956. 22, 27

[45] R. O. Saber and R. M. Murray, "Agreement problems in networks with directed graphs and switching topology," in *Proceedings of the IEEE Conference on Decision and Control*, Dec. 2003, pp. 4126–4132. 22

[46] J. A. Fax and R. M. Murray, "Information flow and cooperative control of vehicle formations," *IEEE Transactions on Automatic Control*, Sept. 2004. 22

[47] W. Ren, R. W. Beard, and D. B. Kingston, "Multi-agent Kalman consensus with relative uncertainty," in *Proceedings of the American Control Conference*, 2005. 22, 23, 24, 26, 34, 38

[48] P. Chandler, S. Rasumussen, and M. Pachter, "UAV cooperative path planning," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2000. 23

[49] T. W. McLain and R. W. Beard, "Coordination variables, coordination functions, and cooperative timing missions," *AIAA Journal of Guidance, Control, and Dynamics*, vol. 28, no. 1, pp. 150–161, Jan. 2005. 23

[50] J. Bellingham, M. Tillerson, A. Richards, and J. P. How, "Multi-task allocation and path planning for cooperating UAVs," in *Cooperative Control: Models, Applications and Algorithms*. Conference on Coordination, Control and Optimization, Nov. 2001, pp. 1–19. 23

[51] R. W. Beard, T. W. McLain, M. Goodrich, and E. P. Anderson, "Coordinated target assignment and intercept for unmanned air vehicles," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 6, pp. 911–922, Dec. 2002. 23

[52] T. McLain and R. Beard, "Cooperative rendezvous of multiple unmanned air vehicles," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2000. 23

[53] W. Ren and R. W. Beard, "Consensus of information under dynamically changing interaction topologies," in *Proceedings of the American Control Conference*, 2004. 24, 27

[54] J. Wolfowitz, "Products of indecomposable, aperiodic, stochastic matrices," *Proceedings of the American Mathematical Society*, vol. 15, pp. 733–736, 1963. 25, 39

[55] W. J. Rugh, *Linear System Theory*. Prentice Hall, 1996, Theorem 6.13, p. 106. 27

[56] ——, *Linear System Theory*. Prentice Hall, 1996, Lemma 12.4, p. 206. 27

[57] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 2002, Lemma 4.7, p. 180. 27

[58] J. Cortés, "Distributed algorithms for reaching consensus on general functions," *Automatica*, (submitted). 35

[59] C. Godsil and G. Royle, *Algebraic Graph Theory*. Springer-Verlag New York, Inc., 2001. 36, 71

[60] 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. 49

[61] E. W. Frew and D. A. Lawrence, "Cooperative stand-off tracking of moving targets by a team of autonomous aircraft," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2005. 55, 56

[62] Z. Tang and U. Ozguner, "Motion planning for multitarget surveillance with mobile sensor agents," *IEEE Transactions on Robotics and Automation*, Oct. 2005. 55, 61

[63] G. Gu, P. R. Chandler, C. Schumacher, A. Sparks, and M. Pachter, "Optimal cooperative sensing using a team of UAVs," *IEEE Transactions on Aerospace and Electronic Systems*, Oct. 2006. 55

[64] R. Sepulchre, D. A. Paley, and N. E. Leonard, "Stabilization of planar collective motion: All-to-all communication," *IEEE Transactions on Automatic Control*, June 2007. 56, 57

[65] D. J. Klein and K. A. Morgansen, "Controlled collective motion for trajectory tracking," in *Proceedings of the American Control Conference*, 2006. 56

[66] G. Yang and V. Kapila, "Optimal path planning for unmanned air vehicles with kinematic and tactical constraints," in *Proceedings of the IEEE Conference on Decision and Control*, 2002. 61

[67] P. R. Chandler, M. Pachter, and S. Rasmussen, "UAV cooperative control," in *Proceedings of the American Control Conference*, 2001. 61

[68] S. R. Griffiths, "Vector field approach for curved path following for miniature aerial vehicles," in *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2006. 61, 63, 64, 66

[69] H. K. Khalil, *Nonlinear Systems*. Prentice Hall, 1996, Theorem 4.4, p. 128. 66

[70] 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. 79, 87, 106, 107

[71] J. Clark and R. Fierro, "Cooperative hybrid control of robotic sensors for perimeter detection and tracking," in *Proceedings of the American Control Conference*, 2005. 83, 85

[72] 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). 83

[73] 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. 83

[74] 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. 83

[75] 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. [Online]. Available: http://www.nosc.mil/robots/land/mdars/auvsmrha.html. 83, 84

[76] S. Young, M. Forshaw, and M. Hodgetts, "Image comparison methods for perimeter surveillance," in *Proceedings of the International Conference on Image Processing and Its Applications*, 1999. 83

[77] J. O. Peralta and M. T. C. de Peralta, "Security PIDS with physical sensors, real-time pattern recognition, and continuous patrol," *IEEE Transactions on Systems, Man and Cybernetics, Part C*, vol. 32, pp. 340–346, Nov. 2002. 83

[78] A. S. Barry and J. Czechanski, "Ground surveillance radar for perimeter intrusion detection," in *Proceedings of the Digital Avionics Systems Conference*, 2000. 83

[79] Space and Naval Warfare Systems Command, "Mobile detection assessment and response system (MDARS)." [Online]. Available: http://www.nosc.mil/robots/land/mdars/mdars.html. 84

[80] M. Kemp, A. L. Bertozzi, and D. Marthaler, "Multi-UUV perimeter surveillance," in *Proceedings of the IEEE/OES Autonomous Underwater Vehicles Conference*, 2004. 85

[81] 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. 85

[82] W. Ren and R. W. Beard, "Consensus seeking in multi-agent systems under dynamically changing interaction topologies," *IEEE Transactions on Automatic Control*, vol. 5, no. 5, pp. 655–661, May 2005. 98

[83] D. B. Kingston and R. W. Beard, "Discrete-time average-consensus under switching network topologies," in *Proceedings of the American Control Conference*, 2006. 99