

**Reaching Consensus with Uncertainty  
on a Network**

by

Cameron S. R. Fraser

Submitted to the Department of Aeronautics and Astronautics  
in partial fulfillment of the requirements for the degree of

Master of Science in Aeronautics and Astronautics

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2009

© Massachusetts Institute of Technology 2009. All rights reserved.

Author .....  
Department of Aeronautics and Astronautics  
August 20, 2009

Certified by .....  
Jonathan P. How  
Professor  
Thesis Supervisor

Accepted by .....  
Professor David L. Darmofal  
Associate Department Head  
Chair, Committee on Graduate Students



# Reaching Consensus with Uncertainty on a Network

by

Cameron S. R. Fraser

Submitted to the Department of Aeronautics and Astronautics  
on August 20, 2009, in partial fulfillment of the  
requirements for the degree of  
Master of Science in Aeronautics and Astronautics

## Abstract

As modern communication networks become increasingly advanced, so does the ability and necessity to communicate to make more informed decisions. However, communication alone is not sufficient; the method by which information is incorporated and used to make the decision is of critical importance.

This thesis develops a novel distributed agreement protocol that allows multiple agents to agree upon a parameter vector particularly when each agent has a unique measure of possibly non-Gaussian uncertainty in its estimate. The proposed *hyperparameter consensus* algorithm builds upon foundations in both the consensus and data fusion communities by applying Bayesian probability theory to the agreement problem. Unique to this approach is the ability to converge to the centralized Bayesian parameter estimate of non-Gaussian distributed variables over arbitrary, strongly connected networks and without the burden of the often prohibitively complex filters used in traditional data fusion solutions. Convergence properties are demonstrated for local estimates described by a number of common probability distributions and over a range of networks. The benefit of the proposed method in distributed estimation is shown through its application to a multi-agent reinforcement learning problem.

Additionally, this thesis describes the hardware implementation and testing of a distributed coordinated search, acquisition and track algorithm, which is shown to capably handle the conflicting goals of searching and tracking. However, it is sensitive to the estimated target noise characteristics and assumes consistent search maps across the fleet. Two improvements are presented to correct these issues: an adaptive tracking algorithm which improves the confidence of target re-acquisition in periodic tracking scenarios, and a method to combine disjoint probabilistic search maps using the hyperparameter consensus algorithm to obtain the proper centralized search map.

Thesis Supervisor: Jonathan P. How  
Title: Professor





## Acknowledgments

First, I would like to thank my advisor, Professor Jonathan How, for the opportunity to pursue my Masters career with the Aerospace Controls Lab, and for his support and guidance over the last two years. I would also like to thank Dr. Luca Bertuccelli, whose knowledge and insights into many facets of this thesis have been invaluable, and Dr. Han-Lim Choi for his assistance with various projects throughout my time here at MIT.

To my friends in the ACL I give many thanks for the helpful advice, camaraderie, and occasional distractions that made working in the lab so enjoyable, even in the wee hours of the morning. In particular, I would like to thank Frant Sobolic and Karl Kulling for their friendship during my time here and with whom I have had the privilege of working side by side through many, many late nights. Thanks to Sergio Cafarelli, Dan Levine, Brandon Luders, Buddy Michini, Brett Bethke, Josh Redding, Frank Fan, Sameera Ponda, and Andy Whitten, who have all been sources of brilliance, perseverance, and energy and were always willing to lend a hand if needed. Finally, I would like to thank Kathryn Fisher for her tireless work on behalf of everyone in the lab.

I would like to give my most heartfelt thanks and appreciation to my friends and family back home for providing me with so much support over the years. To Stephanie McTague, who has stood with me during my time here at MIT, in spirit if not always in person, and who has been a pillar of strength and determination. To my loving parents, who have always given me the courage and inspiration to reach for my dreams, as well as to my Aunt Jan who has been such a beacon of light throughout my life, thanks for always being there for me.

This work was funded by AFOSR # FA9550-08-1-0086



# Contents

<b>1</b>	<b>Introduction</b>	<b>17</b>
1.1	Literature Review . . . . .	22
1.1.1	Consensus . . . . .	22
1.1.2	Data Fusion . . . . .	25
1.2	Contributions . . . . .	27
<b>2</b>	<b>Background</b>	<b>31</b>
2.1	Notation . . . . .	31
2.2	Graph Theory . . . . .	32
2.3	Consensus . . . . .	33
2.3.1	Average Consensus . . . . .	34
2.3.2	Belief Consensus . . . . .	39
2.3.3	Kalman Consensus . . . . .	40
2.4	Summary . . . . .	44
<b>3</b>	<b>Bayesian Hyperparameter Consensus</b>	<b>47</b>
3.1	Derivation . . . . .	48
3.1.1	The Bayesian Kalman Filter . . . . .	49
3.1.2	The Centralized Bayesian Estimate . . . . .	51
3.1.3	The Hyperparameter Consensus Method . . . . .	53
3.2	Illustrative Examples . . . . .	63
3.2.1	The Gamma Prior . . . . .	64
3.2.2	Unknown Network . . . . .	73

3.2.3	The Dirichlet Prior . . . . .	80
3.3	Summary . . . . .	92
<b>4</b>	<b>Application to Cooperative Markov Decision Processes</b>	<b>95</b>
4.1	Markov Decision Processes . . . . .	96
4.1.1	Relationship to Reinforcement Learning . . . . .	98
4.2	Machine Repair Problems . . . . .	101
4.2.1	The Single-Machine Repair Problem . . . . .	101
4.2.2	The Multi-Agent Machine Repair Problem . . . . .	104
4.2.3	The Multi-Agent Machine Repair Problem with Time Considerations . . . . .	117
4.3	Summary and Future Work . . . . .	129
<b>5</b>	<b>CSAT in RAVEN</b>	<b>133</b>
5.1	CSAT and RAVEN . . . . .	134
5.1.1	CSAT . . . . .	135
5.1.2	RAVEN . . . . .	142
5.2	Implementation . . . . .	145
5.3	Results . . . . .	146
5.3.1	Process Noise Adaptation . . . . .	150
5.3.2	Search Map Consensus . . . . .	163
5.4	Summary . . . . .	171
<b>6</b>	<b>Conclusion</b>	<b>173</b>
6.1	Summary . . . . .	173
6.2	Future Work . . . . .	179
<b>A</b>	<b>Bayesian Derivation of the Kalman Filter</b>	<b>181</b>
A.1	Distributed Kalman Filter . . . . .	187
A.2	Kalman Consensus Filter . . . . .	188
	<b>References</b>	<b>189</b>

# List of Figures

1-1	Estimate achieved by average consensus on an arrival rate, $\lambda$ . . . . .	20
1-2	Estimate achieved by Kalman consensus on mean and variance of $\lambda$ . . . . .	21
2-1	Directed, strongly connected five agent network . . . . .	32
3-1	Five agent connectivity graphs . . . . .	70
3-2	Local parameter estimate and variance achieved by hyperparameter consensus on a balanced network . . . . .	71
3-3	Hyperparameter trajectories during consensus on a balanced network . . . . .	71
3-4	Local estimate and variance achieved by hyperparameter consensus on $\lambda$ over a known, unbalanced network . . . . .	72
3-5	Hyperparameter trajectories during consensus on $\lambda$ over a known, unbalanced network . . . . .	72
3-6	Local estimate and variance achieved by hyperparameter consensus for $\lambda$ with a measurement at $k=20$ . . . . .	74
3-7	Local estimate and variance achieved by hyperparameter consensus for $\lambda$ with a measurement at $k=5$ . . . . .	74
3-8	Hyperparameter trajectories during consensus on $\lambda$ over a balanced network with a measurement at $k=5$ . . . . .	75
3-9	Monte-Carlo error results obtained over an unknown network . . . . .	79
3-10	Local parameter estimate achieved for $\mathbf{p}$ on a balanced network . . . . .	86
3-11	Local error in Dirichlet covariance for the hyperparameter and Kalman consensus methods on a balanced network . . . . .	87

3-12	Hyperparameter trajectories during consensus on $\mathbf{p}$ over a balanced network . . . . .	87
3-13	Local estimate achieved by the hyperparameter consensus method on $\mathbf{p}$ over a known, unbalanced network . . . . .	89
3-14	Local error in Dirichlet covariance for hyperparameter consensus on a known, unbalanced network . . . . .	89
3-15	Hyperparameter trajectories during consensus on $\mathbf{p}$ over a known, unbalanced network . . . . .	90
3-16	Local estimate achieved by the average consensus method on $\mathbf{p}$ over a known, unbalanced network . . . . .	90
3-17	Local estimate achieved by hyperparameter consensus on $\mathbf{p}$ with a measurement at $t=20$ . . . . .	91
3-18	Local error in Dirichlet covariance for the hyperparameter consensus over a known, unbalanced network with a measurement at $t=20$ . . .	91
3-19	Hyperparameter trajectories during consensus on $\mathbf{p}$ over a known, unbalanced network with a measurement at $t=20$ . . . . .	92
4-1	Evolution of post-consensus MDP transition probability estimates . .	112
4-2	Probability of obtaining the optimal policy using hyperparameter and average consensus . . . . .	113
4-3	Expected error in the discounted future reward for the working state using hyperparameter and average consensus . . . . .	114
4-4	Expected error in the discounted future reward for the broken state using hyperparameter and average consensus . . . . .	115
4-5	Probability trajectories through probability space with optimal policy basins . . . . .	116
4-6	Evolution of post-consensus MDP transition probability estimates with temporal considerations . . . . .	122
4-7	Evolution of post-consensus $\lambda_{f,r}$ parameters of MDP model . . . . .	123
4-8	Evolution of post-consensus $\lambda_{m,n}$ parameters of MDP model . . . . .	124

4-9	Probability of obtaining the optimal policy using hyperparameter and average consensus with temporal considerations . . . . .	125
4-10	Expected error in the discounted future reward for the working state using hyperparameter and average consensus with temporal considerations . . . . .	126
4-11	Expected error in the discounted future reward for the broken state using hyperparameter and average consensus with temporal considerations	127
4-12	Probability trajectories through probability space with optimal policy basins with different $\lambda$ 's . . . . .	128
5-1	CSAT architecture block diagram . . . . .	136
5-2	RAVEN architecture . . . . .	143
5-3	Modified Hummingbird UAVs performing a coordinated search and track task on tank targets . . . . .	144
5-4	CSAT mission performance . . . . .	147
5-5	UAV mode and search area comparison for different $Q$ values . . . . .	148
5-6	Revisit time vs $Q$ , parameterized by sensor size . . . . .	155
5-7	Revisit time vs $Q$ , parameterized by sensor noise . . . . .	156
5-8	$\hat{Q}$ estimation trajectories with varying gains . . . . .	157
5-9	Percentage of targets lost using adaptive vs non-adaptive tracking . . . . .	159
5-10	$Q$ -adaptation with different measurement capabilities . . . . .	161
5-11	Response to a step change in $Q$ . . . . .	162
5-12	Static consensus on search maps . . . . .	168
5-13	Evolution of errors in the search map . . . . .	169
5-14	Evolution of errors in the search map with sparse communication . . . . .	170





# List of Tables

2.1	Graph theory notation . . . . .	31
2.2	Consensus notation . . . . .	32
3.1	Properties of parameterized distributions relating to consensus . . . . .	65
3.2	Initial conditions for $\lambda$ consensus . . . . .	69
3.3	Expected consensus error comparison with initial conditions . . . . .	78
3.4	Initial conditions for $\alpha$ consensus . . . . .	84



# List of Algorithms

1	Hyperparameter consensus method with globally shared information .	60
2	Hyperparameter consensus method with local measurement updates .	63
3	Revisit time and location calculation . . . . .	140
4	1-D revisit time and location calculation . . . . .	154



# Chapter 1

## Introduction

In recent years, the development and utilization of quick, reliable communication networks has shrunk the world and allowed for collaboration between all corners of the globe. People and computers are able to communicate across great distances to share thoughts, ideas, and information, achieving a greater collective understanding of the world and facilitating more informed decisions. From medical databases to scientific collaboration to military information collection, these communication networks are becoming more integrated with modern life and are continually improving their abilities and scope to match the growing demand. Currently, doctors can query a patient's database to get past health information, combine it with test results accessed online from a clinic's repository, and confer with experts in other hospitals in order to make an informed decision, even when all these people and locations are distributed around the country. Alternately, field military commanders can receive camera images of an enemy target from a reconnaissance drone, locations of nearby troops from global positioning satellites, and plan with other commanders prior to finalizing any troop movement.

Many of these systems can be generalized as a collection of possibly heterogeneous nodes communicating with each other over a connectivity graph. The doctor, database, clinic, and other experts all need to be consulted before making an important diagnosis, just as the commander, drone, and other personnel are all nodes in the military network that have information relevant to completing their objective.

The end users of this information often need to make decisions based on the available knowledge and, further, often need to be in agreement with other similar agents on the same network. For example, multiple field commanders all need to be in agreement on the state of the world in order to make consistent, coherent, and coordinated plans. If one commander does not have the same knowledge as the others, they may decide on plans that are redundant or possibly catastrophic, throwing the entire mission into jeopardy. This problem of agreement across networks of agents is the primary focus of the field of consensus. Whether it is agreement between friends on when to meet for dinner, between investors on the riskiness of a purchase, or between commanders on the state of the battlefield, intelligent and coordinated decision-making requires agreement [1].

In particular, the increasing autonomy of unmanned vehicles is making it possible for some decisions to be made independently of human operators. Because of their scalability and robustness to individual failure, it is of great interest to have “swarms” of unmanned aerial vehicles (UAVs) that can autonomously search for and track targets in regions of interest, allowing fewer human operators to oversee larger groups of assets, therefore saving time and money. The performance of this task, however, is often greatly dependent on coordination through shared information. For example, formation flying requires all agents to have knowledge of the positions of the other agents and the desired formation shape - if one agent is not in agreement about the desired heading, the formation dissolves.

Currently, consensus methods exist for many different applications. In particular, coordinated control of vehicles, such as UAVs in formation and multi-agent rendezvous, has received much recent attention. These methods generally focus on agreement on some consensus parameter, such as heading angle or rendezvous time in the previous examples, and seek to come to a (possibly weighted) average of the agent’s initial estimates. This class of consensus is often called *average consensus* (AC) [2, 3]. Once a consistent value has been reached across the fleet, each agent can then plan locally to remain in its desired state. This problem is often highlighted by the “meet for dinner” example introduced in [1]:

A group of friends have decided to go out for dinner tonight, but fail to specify a time to meet. Over the course of the afternoon, each friend realizes this dilemma and must get in contact with the other friends to agree upon a time. If a conference call is an option then the friends could debate quickly and find a suitable answer, but, unfortunately, this only shifts the focus from what time to have dinner to what time to have the call. Instead, each friend must call the others individually, combine information locally, and update their current estimate of the time to meet. Once a time has been agreed to by all the friends, they can each go about planning the rest of their day so that they successfully meet for dinner at night.

This example is very simple and represents the motivation behind the majority of consensus methods. However, it lacks the ability to consider any confidence or uncertainty associated with each friend's estimate, as may arise in the following situation:

Suppose, recalling their problem from last time, the same group of friends this time visits the restaurant a day in advance and make a reservation as a group. However, each friend is preoccupied in various forms and some do not pay full attention as the reservation is made, such that, after they part ways, each person has a different idea of when to meet for dinner as well as some confidence in their belief. In this case, the friends are faced with a similar problem, but now must also try and maximize their collective confidence in their estimates in the hope of arriving on time.

This type of consensus requires taking into account a richer form of the agents' information, usually represented by uncertainties in the estimate. These uncertainties can arise through subjective or objective means, such as prior knowledge, differences in experience or number of measurements, or varying sensor qualities. Figure 1-1 shows an example of the discrepancy that could be obtained if confidences are ignored and agents simply agree upon an average of their mean estimates. The true, centralized

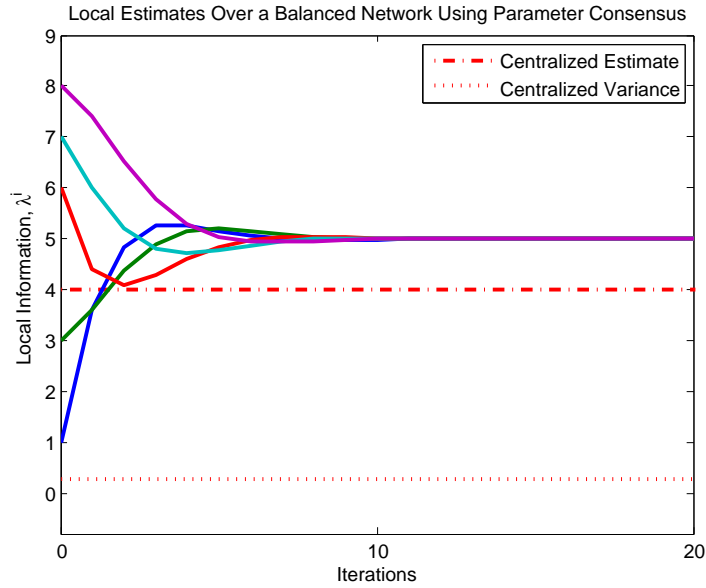


Figure 1-1: Estimate achieved by average consensus on an arrival rate,  $\lambda$ . Solid lines represent each agent’s belief; dash-dot lines represent the proper centralized estimate achieved using a Bayesian combination of each agent’s uncertain estimate

values shown via dashed lines are obtained through a Bayesian aggregation of the local beliefs with uncertainties.

Some methods, such as those inspired by the Kalman filter, have been developed to take into account uncertainties in the agents’ respective measurements, but are only accurate in certain situations. For example, Figure 1-2 shows the result of a *Kalman consensus* [4, 5] (KC) approach utilizing the mean and variance of each agent’s initial estimate. The steady-state values for both the mean and variance (solid and dashed lines, respectively) are significantly different from the Bayesian result, suggesting that the Kalman approach is not accurate in this scenario. Further, even in the linear-Gaussian uncertainty framework that the KC algorithm was designed for, the resulting consensus estimate is guaranteed to converge to the centralized Bayesian estimate but the resulting variance may be badly biased.

Belief consensus (BC) methods [6] consider uncertainties in a different manner by discretizing the possible values of the variable of interest and then coming to consensus on the likelihood of each outcome. This method can be thought of as an extension



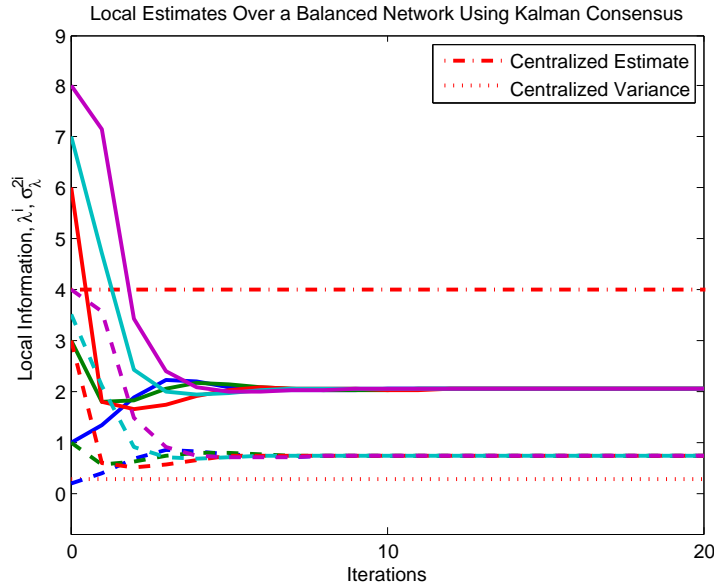


Figure 1-2: Estimate achieved by Kalman consensus on mean and variance of  $\lambda$ . Solid lines represent each agent’s mean belief, dashed color lines represent the agent’s associated uncertainty in their estimate; dash-dot and dotted lines represent the proper centralized mean estimate and variance, respectively, achieved using a Bayesian combination of each agent’s uncertain estimate

to the second meet-for-dinner problem where, for example, the available times are discretized to fall on each quarter-hour from 6:30 to 7:30. Each agent then maintains a likelihood of each time being correct, and the time with the highest combined likelihood is then considered the most probable. Though useful for hypothesis testing situations, it is undesirable in the generic case since the discretization limits the range of values allowed, and it does not consider the possibility of the likelihoods themselves to be uncertain.

These three methods are the primary protocols in the consensus community, as will be discussed next in the Literature Review, but they all have limitations when uncertainties factor into the decision. The AC protocols do not explicitly consider uncertainties in the parameter, while KC methods only accurately capture the uncertainties in select situations. BC allows for variability of a parameter over a set of possible outcomes, but does not consider uncertainties in the local beliefs. This improper consideration of uncertainties can not only lead to badly biased consensus

values but can also affect performance in situations that are sensitive to the parameter in question. Further, some recent investigations have highlighted the importance of robustness in decision making through taking uncertainties explicitly into account [7]. Thus, the ability to come to agreement on the proper estimate while maintaining a valid measure of uncertainty on the estimate is of critical importance to future control algorithms and has yet to be addressed in the consensus community.

In a manner similar to the BC method, uncertainties in a parameter can be modeled as a probability distribution over a set of allowed values. However, instead of discretizing the domain, if the entire distribution can be communicated and agreed upon then a much more robust, adaptable consensus can be achieved. Thus, the goal of this thesis is to develop a consensus method that allows multiple agents to come to agreement upon a probability *distribution* in order to facilitate informed, coordinated, and robust decision-making.

## 1.1 Literature Review

### 1.1.1 Consensus

The modern consensus methods are rooted in the fields of mathematics, computer science [8], and management science [9–11], and have since been adapted to fit into the cooperative control framework. Many consensus algorithms are postulated as differential or difference equations, which have been studied extensively by the mathematics community. Additionally, the Bayesian principles frequently used in this thesis arose initially through study of the theory of probability and statistics, and have been adopted by the controls community through estimation methods such as the well-known Kalman filter [12].

#### Early Work

Early investigations of consensus as an agreement strategy are drawn from the management science and statistics literature. DeGroot and his contemporaries (see [9]

and references therein) are concerned with how to combine many ‘expert opinions’ into a single, central representation that can be used, for example, in risk assessment. In [10], Winkler compares different centralized methods to combine a group of subjective probability distributions and demonstrates the dependence of the final consensus value on the particular method used, but makes no claim to the *correctness* of the examined Bayesian or Kriged (weighted-average) solutions. Further, these initial investigations consider entirely centralized approaches where all the required information is available to a single decision-making agent, and neglects any communication framework required to achieve these ends.

Many others extended this work to evaluate the asymptotic convergence of shared probabilistic Bayesian beliefs [13–17]. Primary focus was given to the question of the existence of a centralized estimate in the two-agent case, but without any explicit mention of how, exactly, to communicate each agent’s knowledge. Despite this, a straightforward but very important result was given by Aumann in [13], where he states that if two agents have the same prior belief and all subsequent knowledge about an event is considered “common knowledge”, their posterior beliefs accounting for all shared information must be equivalent. This result is fundamental in the problem of agreement on distributions, but provides little insight on how to actually share common knowledge.

### **Average Consensus**

The motivation of most consensus methods considered in this thesis is the need for agreement in the context of coordinated control, with primary focus on flocking. Initially, Reynolds [18] proposed a model of biological flocking for use in computer simulations that was described as an emergent behavior achieved by individual agents through comparing their current state with their neighbors’ and adjusting according to some local weights. Vicsek *et al.* [19] formalized this flocking model by showing through simulation that, if each agent adjusts its heading to the average heading of agents close to it, the agents will converge locally to identical headings.

Vicsek’s model has motivated much work on the question of consensus for applica-

tions to distributed control of autonomous systems [1–3, 20–30]. Jadbabaie *et al.* [20] studied the problem of autonomous flocking through heading alignment and showed that agents were able to converge to an equilibrium heading value using undirected, nearest-neighbor communication graphs so long as the graphs were connected “frequently enough”. A formal study of the information-theoretic approach to consensus was conducted in [21], which proposed a linear consensus filter to achieve consensus on a generic piece of information over a network, leading to the general class of *average consensus* methods. A generic analysis of consensus variables and methods was conducted in [1], while many others have extended the results in [20, 21] to the case of consensus over directed, fixed or time-varying networks [23, 25, 26] and networks to promote high convergence speeds [27]. Most consensus methods utilize extensions of the average consensus method (see [2, 3]), such as the dynamic-average consensus filters in [31, 32] and belief consensus algorithm in [6].

## Belief Consensus

Olfati-Saber *et al.* [6] introduced the concept of *belief consensus* as a scalable implementation of the decentralized Bayesian identification method proposed by Rao and Durrant-Whyte [33]. The goal of belief consensus is to combine the estimates of many distributed sensors through consensus on a product of likelihood estimates to be used for distributed hypothesis testing, such as object classification. This approach allows for a distributed set of nodes to agree on the most likely hypothesis out of a predefined set of possible outcomes.

In addition to classification, it can be utilized for estimation of a parameter of interest that is confined to a finite set of possible outcomes. However, belief consensus is unfortunately limited to these discrete settings and is not applicable to estimation of continuous variables. Further, though it does allow for a measure of uncertainty through consideration of multiple hypotheses, belief consensus does not consider uncertainties in the local beliefs themselves, which can lead to incorrect classifications based on the observed data.

## Kalman Consensus

In order to consider uncertainties in local estimates, some have approximated the consensus problem with a multi-sensor Kalman filter [4, 5]. This class of Kalman consensus filters was originally introduced in [4], where each agent not only has an estimate of the consensus value but an associated variance in its belief. Assuming that the estimates are Normally distributed, the consensus algorithm is formulated to consider communication of each agent’s local estimate and variance as if it is a measurement made from the process trying to be estimated. This repeated integration of *pseudo-measurements* from the other agents allows the agents to apply traditional Kalman filter measurement updates to information received from neighboring nodes. This allows the agents to arrive at a consensus value that not only takes into account all the agents’ initial estimates but also their uncertainties. This method was shown to be subject to network-induced biases by [5], who extended the algorithm to converge to the proper unbiased estimate (under the same linear-Gaussian assumption) over unbalanced networks through adjusting the edge weights of the communication graph. Though this broadened the application of the Kalman consensus algorithm to a wider class of network topologies, the fundamental assumption of Normally distributed local uncertainties still prevents the Kalman consensus algorithm from reaching an unbiased solution on any more general classes of uncertainty distributions.

### 1.1.2 Data Fusion

The goal of this thesis is very closely related to that of the data fusion community [34–42], insofar as the motivation is for information to be aggregated in a well-defined, principled way. This thesis extends concepts in [39, 40] to help combine these two communities through the application of consensus methods to data fusion problems, or, more precisely, through application of data fusion techniques to the consensus framework.

Data fusion is often concerned with multiple sensor fusion or distributed decision making [42], where multiple sensors are making measurements that need to be com-

bined before they can be analyzed and used to aid in the decision-making process. Often, data fusion methods are centralized [42], though many researchers have explored the decentralized problem [34–38], while others have examined the distributed problem [39, 40]. The primary difference between the decentralized and distributed approach is *scalability* of the proposed algorithm [40], described next.

Communication protocols in the decentralized data fusion problems generally rely on the transmission of each node’s complete, current, processed estimate [34–38], which requires the removal of shared or mutual information before inclusion in another agent’s estimate. This mutual information could be due to network-induced redundancies, where an agent receives the same information from multiple sources, as well as through possible correlation of measurements. The latter is often avoided through assuming independence of measurements (an assumption made in this thesis as well), though the former problem still remains. This has motivated the development of *channel filters* that maintain estimates of mutual information between all neighbors of a node such that only the new information is extracted and included in the local estimate. However, in even marginally complex networks where there are multiple paths from one agent to another, the calculation of the mutual information becomes exceedingly complex. Thus, channel filters are very difficult to formulate except on the simplest of networks and make the decentralized data fusion problem very difficult to scale [35, 36].

On the other hand, distributed data fusion problems in [39, 40, 43] make use of consensus-inspired communication protocols to achieve better scalability of the algorithm and avoid the necessity of channel filters for network-induced redundancies. This thesis takes an approach similar to Xiao *et al.* [39], which uses average-consensus methods for communication of local estimates to achieve convergence to the centralized estimate. However, they derive their distributed maximum likelihood estimator assuming linear local measurements of a static process corrupted by Gaussian sensor noise, which results in a distributed representation of the Kalman filter and is applicable only to Normally distributed estimates. Similarly, Olfati-Saber [40] developed an approximate distributed Kalman filtering formulation using dynamic-average con-

sensus filters [43] for communication between local *micro*-Kalman filters and shows the ability of a system with 200 nodes to track a time-varying signal. The use of average and dynamic-average consensus algorithms in these two approaches avoids the necessity for channel filters and allows the system to scale to large, complex networks, though both results remain tied to Normally distributed uncertainties.

In an effort for greater generality, Makarenko and Durrant-Whyte [38] propose a generic Bayesian decentralized data fusion framework for arbitrary distributions, but the proposed architecture still heavily relies on channel filtering and is, again, only demonstrated for the Kalman case. Thus, while the data fusion community is focused on the proper aggregation of locally uncertain data, almost all approaches for distributed estimation utilize the Kalman filter framework. Further, those methods that do support generic distributions are still heavily confined by reliance on channel filters. Thus, it has been shown that the application of consensus methods for increased scalability of non-Normally distributed information is a relatively unexplored field in both the consensus and data fusion communities which this thesis will seek to address.

## 1.2 Contributions

The primary contribution of this thesis is the derivation and application of the hyperparameter consensus method to address some of the primary shortcomings in the current consensus and data fusion literature; in particular, the lack of a scalable algorithm to achieve an unbiased consensus with non-Normally distributed local parameter uncertainties. A secondary contribution of the thesis is two methods for improving performance of an autonomous multi-agent coordinated search, acquisition, and track problem, partially aided by results from the hyperparameter consensus method. With these goals in mind, the thesis will be proceed as follows:

Chapter 2 identifies and defines the inability of modern consensus methods to accurately handle many parameterized uncertain beliefs in agents' initial conditions as well as the lack of any consensus method to agree on probability distributions

themselves. Additionally, we recognize that Bayesian data fusion techniques exist to address these problems, but they generally rely on complex message-passing and channel filtering schemes to share information. The proposed approach relies on fundamental consensus protocols defined in this chapter to achieve the same ends as the data fusion approach, but without requiring such a complex communication structure.

The hyperparameter consensus method introduced in Chapter 3 merges these two fields by applying Bayesian probability theory to the consensus problem. This combination allows multiple agents to come to a distributed agreement not only on the centralized Bayesian parameter estimate but also on a range of parameterized distributions themselves, particularly when uncertainties are present in the local parameter estimates. A key factor is that the Bayesian approach provides a framework upon which existing linear consensus algorithms can factor in local uncertainties in a meaningful way, while maintaining the flexibility inherent in these protocols. Further, the convergence properties of the consensus protocols negate the requirement for the complex message routing and channel filtering schemes of the data fusion community.

Some particular contributions of the hyperparameter consensus method to various fields are given below:

- Contributions to the consensus community:
  - A new consensus algorithm that permits more robust and accurate decision-making by allowing for multiple agents to converge on the centralized Bayesian parameter estimate given uncertain, not necessarily Gaussian, local information.
  - The application of Bayesian probability theory to obtain a method by which agents can come to an unbiased agreement on an entire parameterized distribution through a proper consensus on its parameters.
- Contributions to the data fusion community:
  - Application of a consensus-inspired communication scheme on the hyper-



parameters of local, non-normal prior and posterior probability distributions to simplify communication and relax the requirement for channel filters.

- Contributions to the cooperative control community:
  - Application of the hyperparameter consensus method to the estimation and communication of transition probabilities for observed discrete event systems.
  - A more accurate means by which agents can simultaneously learn and communicate possibly uncertain decision parameters in order to achieve better performance using implicit coordination techniques.

Chapter 4 demonstrates the performance of the hyperparameter consensus algorithm in a multi-agent learning context using Markov Decision Processes. It is shown that this method allows for greatly expedited convergence to estimated model parameters through the Bayesian aggregation of multiple independent, uncertain local estimates. This allows multiple agents locally observing independent and identically distributed processes to converge much faster to the true model parameters through utilization of the increased observational power of the team.

Finally, the Chapter 5 presents the hardware implementation and testing of a coordinated search, acquisition, and track mission management algorithm, as well as some subsequent research motivated by the observed results. The mission scenario considered here is an ongoing area of research in the coordinated planning and control community, for which the presented hardware implementation is among the first of its kind. The algorithm itself combines the competing goals of searching and tracking in a synergistic framework through the proper scheduling of agents to the respective duties, particularly in resource-deficient scenarios. Sensitivities in the algorithm observed during testing motivated the investigation of two performance enhancing modifications: first, adaptive modeling of track targets in order to improve the quality of the allocation of resources to each task. It is shown through simplified simulation results that adaptation of the effective process noise of the target model

can improve the resource allocation problem. This is done by maximizing the time to search unknown territory while still maintaining a set confidence bound on the re-acquisition of a temporarily un-observed target. Second, an application of hyperparameter consensus to agreement on search maps is introduced to mitigate errors in situational awareness accrued while agents are searching but out of communication with the rest of the fleet.

The last chapter provides some concluding thoughts and discussion about the demonstrated algorithms and results. A brief future work section outlines some of the primary avenues of future research to expand the application of the hyperparameter consensus method.

# Chapter 2

## Background

This chapter introduces the required background in current consensus methods to properly formulate and motivate the development of the hyperparameter consensus method. In particular, it will introduce the notation to be used throughout the thesis, provide a brief introduction to graph theory, and, expand upon the theory of the current state of the art in three primary consensus methods: average consensus (AC), belief consensus (BC), and Kalman consensus (KC). The protocols used for each method will be defined in order to provide further context for the limitations stated in Chapter 1 and facilitate the derivation in Chapter 3.

### 2.1 Notation

Table 2.1: Graph theory notation

$\mathcal{V}$	Set of nodes in a graph
$\mathcal{E}$	Set of edges connecting nodes, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$
$\mathcal{G} = [g_{ij}]$	Connectivity graph, $g_{ij} = \begin{cases} 1 & \text{if } (j, i) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases} \quad \forall i, j \in \mathcal{V}$
$D = [a_{ij}]$	<i>Adjacency</i> matrix associated with $\mathcal{G}$ , $a_{ij} \begin{cases} \geq 0 & \text{if } (j, i) \in \mathcal{E} \\ = 0 & \text{otherwise} \end{cases} \quad \forall i, j \in \mathcal{V}$
$\nu$	Consensus eigenvector of $D$

Table 2.2: Consensus notation

$\xi$	Consensus information vector
$\mathbf{w}$	Vector of weights
$(\cdot)_i$	Value associated with node $i \in \mathcal{V}$
$(\cdot)[k]$	Value associated with consensus iteration $k$
$(\cdot)^*$	Steady state or consensus value
$\text{diag}(\cdot)$	Diagonal matrix with diagonal entries defined by the argument
$(\cdot)^\dagger$	Element-wise inverse operator
$\odot$	Element-wise multiplication operator
$\mathbf{e}$	Column vector of 1's

## 2.2 Graph Theory

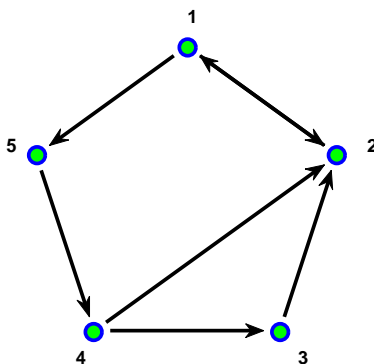


Figure 2-1: Directed, strongly connected five agent network

Let  $\mathcal{V}[k] = \{1, \dots, N[k]\}$  denote a set of  $N[k]$  vertices at consensus iteration  $k$ . The set of edges,  $\mathcal{E}[k] \subseteq \mathcal{V}[k] \times \mathcal{V}[k]$ , is defined by pairs  $(i, j) \in \mathcal{E}[k]$  if and only if node  $i$  can talk to node  $j$  at time  $k$ . It is assumed that each agent is able to talk to itself at all times  $((i, i) \in \mathcal{E}[k] \forall i, k)$ . The set of vertices and edges defines a graph,  $\mathcal{G}[k] = (\mathcal{E}[k], \mathcal{V}[k]) = g_{ij}[k]$ , where

$$g_{ij}[k] = \begin{cases} 1 & \text{if } (j, i) \in \mathcal{E}[k] \\ 0 & \text{otherwise} \end{cases} \quad \forall i, j \in \mathcal{V}[k]$$

An adjacency matrix,  $D[k]$ , can be defined for each graph and will be used to describe the impact of one agent's information on another. It is composed of elements  $a_{ij}[k]$  that represent the weight agent  $i$  gives to information from agent  $j$  (the weight

of edge  $(j, i)$ ), and follows

$$a_{ij}[k] \begin{cases} \geq 0 & \text{if } (j, i) \in \mathcal{E}[k] \\ = 0 & \text{otherwise} \end{cases} \quad \forall i, j \in \mathcal{V}[k]$$

A graph is considered undirected if  $g_{ij} = 1 \Leftrightarrow g_{ji} = 1$ , and directed otherwise. A further assumption on undirected graphs is that the edge weights,  $a_{ij}$ , are equal in each direction,  $a_{ij} = a_{ji}$ . A directed path from  $i$  to  $j$  is a sequence of directed edges starting at node  $i$  and ending at  $j$ ,  $(i, i_1)(i_1, i_2) \dots (i_m, j)$ . A graph is considered *strongly connected* if there exists a directed path from every node to every other node (eg. Figure 2.2). The adjacency matrix  $D$  is called *balanced* if  $\sum_{j=1}^N a_{ij} = \sum_{j=1}^N a_{ji} \forall i$ , or, in other words, if the weight of all incoming information to a node is equal to the weight of all outgoing information from that node (note that this condition is automatically satisfied if the graph is undirected, and is also equivalent to  $D$  being doubly-stochastic, where all row- and column-sums are unity).

## 2.3 Consensus

The modern consensus problem is generally considered that of obtaining agreement across a network of agents to a common value of a parameter of interest. This section will formally define the three primary protocols of interest to this thesis: average consensus, belief consensus, and Kalman consensus. The discussion to follow will focus on *discrete time* consensus protocols due to their direct applicability to computer systems, though the following methods also have analogous continuous time equivalents. The AC algorithms will be described first, which achieve agreement to the arithmetic average of the initial conditions. Next, the BC will be discussed as it pertains to distributed hypothesis testing, and, finally, the KC algorithm will be introduced, which assumes a level of uncertainty in the local knowledge and utilizes this to bias the consensus value towards the more confident agents.

### 2.3.1 Average Consensus

Let each agent,  $i$ , be represented by a node in a time-invariant communication graph,  $\mathcal{G}$ , where  $i$  is in the set of vertices,  $\mathcal{V} = \{1, \dots, N\}$ ,  $i$  can transmit to some other node  $j$  if and only if  $(i, j)$  is in the set of edges,  $\mathcal{E}$ , and it is implied that an agent can talk to itself:  $(i, i) \in \mathcal{E}$ . It is assumed, for simplicity, that the agents communicate using a synchronous communication protocol over a given connectivity graph that is fixed and strongly connected. The discrete-time updates for a consensus variable  $\xi$  are of the form

$$\xi_i[k+1] = \sum_{j=1}^N a_{ij} \xi_j[k] \quad \forall i \in \mathcal{V} \quad (2.1)$$

In matrix form, this becomes

$$\xi[k+1] = D\xi[k] \quad (2.2)$$

where the edge weights,  $a_{ij}$ , are found explicitly from Eq. 2.1. The resulting adjacency matrix  $D$  is assumed to be non-negative (all  $a_{ij} \geq 0$ ) and row-stochastic ( $\sum_{j=1}^N a_{ij} = 1 \forall i$ ), which is achieved by constraining all non-zero entries to

$$0 < a_{ij} < \frac{1}{\sum_{k=1, k \neq i}^N g_{ik}} \quad (2.3)$$

and

$$a_{ii} = 1 - \sum_{j \neq i} a_{ij} \quad (2.4)$$

Gershgorin's disc theorem can be used to show that all of the eigenvalues of a row-stochastic matrix are within the unit disk with at least one eigenvalue at 1 [44]. If  $D$  has a simple eigenvalue at 1, then there exists a normalized left eigenvector  $\nu$  associated with the 1-eigenvalue of  $D$ , such that

$$\lim_{k \rightarrow \infty} D^k = \mathbf{e}\nu^T \quad (2.5)$$

where  $\mathbf{e}$  is a column vector of ones. This result is guaranteed if the matrix  $D$  is *irreducible* (or, equivalently, if the underlying graph  $\mathcal{G}$  is strongly connected)<sup>1</sup> through invoking the Perron-Frobenius theorem [45]. Thus, the final consensus value is obtained as

$$\lim_{k \rightarrow \infty} \xi_i[k] = \nu^T \xi[0]$$

This result shows that the consensus value is guaranteed to be a convex combination of the initial conditions with weights  $\nu$ , hereafter called the *consensus eigenvector* of  $D$ . It is important to note that  $\nu$  is dependent only on the communication graph  $\mathcal{G}$  and not on the specific values of the  $a_{ij}$ , as long as the properties in Eq. 2.4 and 2.3 are true<sup>2</sup>. If the network is known, each agent can determine its local influence in the resulting consensus value,  $\nu_i \xi_i[0]$ .

In order to achieve consensus to the arithmetic average of the initial conditions, it is sufficient to run the protocol in Eq. 2.2 in a network that gives  $\nu = \mathbf{e}/N$ . If the network is balanced then this condition will be satisfied; however, if the network is not balanced, the consensus eigenvector will not equal  $\mathbf{e}/N$  and the arithmetic average will not be the default steady-state value. In many cases it may be desirable to achieve a particular result (ie. the arithmetic average) from the consensus algorithm. For example, in flocking it may be preferable to converge to the arithmetic average of all initial headings and velocities to minimize the overall alignment work required to be done by the flock. In the case of an unbalanced network, it is possible to combat this network-induced bias by a proper weighting of the initial conditions. Therefore, if each agent knows the influence of its information, given by  $\nu_i$ , their initial conditions can be weighted such that the consensus artificially converges to a desired result.

Given some desired weighted sum of initial conditions with weights defined by  $\nu_{desired}$ , by selecting weights as

$$\mathbf{w} = \nu_{desired} \odot \nu^\dagger \tag{2.6}$$

---

<sup>1</sup>These are sufficient but not necessary conditions for 1 to be a simple eigenvalue.

<sup>2</sup>While the exact edge weights do not factor into the steady-state value, they do, however, play a large role in convergence speed [3].

and applying these weights to the initial conditions  $\xi_i[0^-]$  as

$$\xi_i[0] = w_i \xi_i[0^-]$$

the consensus method in Eq. 2.2 will achieve the desired consensus value of  $\nu_{desired}^T \xi[0^-]$ . For example, with the vector  $\nu_{desired} = \mathbf{e}/N$ , the resulting consensus with the prescribed weights will be the desired average value:

$$\begin{aligned} \lim_{k \rightarrow \infty} \xi_i[k] &= \nu^T \text{diag}(\mathbf{w}) \xi[0^-] \\ &= \sum_{j=1}^N \nu_j w_j \xi_j[0^-] \\ &= \sum_{j=1}^N \nu_j \left( \frac{1}{N \nu_j} \right) \xi_j[0^-] \\ &= \sum_{j=1}^N \frac{1}{N} \xi_j[0^-] \end{aligned}$$

where  $\text{diag}(\mathbf{w})$  denotes the diagonal square matrix with diagonal entries composed of the elements of  $\mathbf{w}$ :

$$\text{diag}(\mathbf{w}) = \begin{bmatrix} w_1 & 0 & \cdots & 0 \\ 0 & w_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_N \end{bmatrix}$$

Finally, if the weights  $w_i$  are selected as  $1/\nu_i$  (ie.  $\nu_{desired} = \mathbf{e}$ ), then the resulting consensus will be a *sum*-consensus, with agreement to the sum of all the initial conditions.

These results can also be extended to some time-varying networks. This is an important extension in cases with limited connectivity due to line-of-sight or distance constraints among dynamic agents or losses due to communication noise. Convergence using Eq. 2.2 over switching topologies has been studied in [20, 23, 25, 26], though, in general, no guarantees about the convergence value are made other than it is within the convex hull of the initial conditions (ie.  $\nu$  exists but is undefined in closed



form). However, in the case of a finite number of switches between known connectivity graphs, the following proposition is true:

**Proposition 2.3.1.** *The consensus protocol in Eq. 2.2 will converge to the desired result*

$$\lim_{k \rightarrow \infty} \xi[k] = \mathbf{e} \nu_{desired}^T \xi[0^-]$$

over a switching network composed of  $K$  switches among known graphs  $\mathcal{G}_0, \mathcal{G}_1, \dots, \mathcal{G}_{K-1}$  when applying local weights after each switch, where the switch from  $\mathcal{G}_{i-1}$  to  $\mathcal{G}_i$  happens at iteration  $k_i$ , with  $k_0 = 0$ .

*Proof.* For each graph,  $\mathcal{G}_i$ , let the associated row-stochastic adjacency matrix be  $D_i$  with consensus eigenvector  $\nu_i$ . The weights will be applied on each switching iteration, such that the weights at iteration  $k_i$  are defined as be  $\mathbf{w}[k_i]$ . The consensus protocol with local weighting is then given as

$$\xi[k+1] = \begin{cases} D_i \text{diag}(\mathbf{w}[k]) \xi[k] & \text{if } k = k_i \\ D_i \xi[k] & \text{for } k_i < k < k_{i+1} \\ D_{K-1} \xi[k] & \text{for } k > k_{K-1} \end{cases}$$

The resulting update at time  $k \in [k_i, k_{i+1})$  is then:

$$\xi[k+1] = D_i^{k-k_i+1} \text{diag}(\mathbf{w}[k_i]) D_{i-1}^{k_i-k_{i-1}} \text{diag}(\mathbf{w}[k_{i-1}]) \dots D_0^{k_1} \text{diag}(\mathbf{w}[0]) \xi[0^-]$$

If the weights are selected according to

$$\mathbf{w}[k_i] = \nu_{i-1} \odot \nu_k^\dagger \tag{2.7}$$

with the initial weighting defined as

$$\mathbf{w}[0] = \nu_{desired} \odot \nu_0^\dagger, \tag{2.8}$$

then the information at time  $k + 1$  can be defined as

$$\xi[k + 1] = D_i^{k-k_i+1} \text{diag}(\nu_{i-1} \odot \nu_i^\dagger) \dots D_0^{k_1} \text{diag}(\nu_{desired} \odot \nu_0^\dagger) \xi[0^-].$$

As  $k \rightarrow \infty$ , the resulting consensus converges to:

$$\begin{aligned} \lim_{k \rightarrow \infty} \xi[k + 1] &= \lim_{k \rightarrow \infty} D_{K-1}^{k-k_{K-1}+1} \text{diag}(\nu_{K-2} \odot \nu_{K-1}^\dagger) \dots D_0^{k_1} \text{diag}(\nu_{desired} \odot \nu_0^\dagger) \xi[0^-] \\ &= \mathbf{e} \underbrace{\nu_{K-2}^T \text{diag}(\nu_{K-2} \odot \nu_{K-1}^\dagger)}_{\nu_{K-2}} D_{K-2}^{k_{K-1}-k_{K-2}} \text{diag}(\nu_{K-3} \odot \nu_{K-2}^\dagger) \dots \\ &\quad \dots D_0^{k_1} \text{diag}(\nu_{desired} \odot \nu_0^\dagger) \xi[0^-] \end{aligned} \quad (2.9)$$

$$\begin{aligned} &= \mathbf{e} \underbrace{\nu_{K-2}^T D_{K-2}^{k_{K-1}-k_{K-2}} \text{diag}(\nu_{K-3} \odot \nu_{K-2}^\dagger)}_{\nu_{K-2}} \dots \\ &\quad \dots D_0^{k_1} \text{diag}(\nu_{desired} \odot \nu_0^\dagger) \xi[0^-] \end{aligned} \quad (2.10)$$

⋮

$$\begin{aligned} &= \mathbf{e} \underbrace{\nu_0^T D_0^{k_1}}_{\nu_0} \text{diag}(\nu_{desired} \odot \nu_0^\dagger) \xi[0^-] \\ &= \mathbf{e} \underbrace{\nu_0^T \text{diag}(\nu_{desired} \odot \nu_0^\dagger)}_{\nu_{desired}^T} \xi[0^-] = \mathbf{e} \nu_{desired}^T \xi[0^-] \end{aligned}$$

where the simplification in (2.9) is because

$$\nu_{K-1}^T \text{diag}(\nu_{K-2} \odot \nu_{K-1}^\dagger) = \nu_{K-1}^T \odot (\nu_{K-2} \odot \nu_{K-1}^\dagger)^T = \nu_{K-2}^T$$

and the simplification in (2.10) is a result of the property of eigenvalues,

$$\nu_i^T D_i^k = \underbrace{\lambda}_1 \nu_i^T D_i^{k-1} = \dots = \nu^T$$

The final equality is the desired result, which completes the proof.  $\square$

### 2.3.2 Belief Consensus

Hypothesis testing is a key component of the data and sensor fusion community, whereby multiple sensor measurements are used to identify or classify an object into one of a predefined set of categories. The belief consensus method by Olfati-Saber *et al.* [6] is an algorithm for hypothesis testing in a distributed setting, where multiple agents attempt to agree on the most likely classification of an event. It builds off of concepts of Bayesian inference utilized in [33] for decentralized Bayesian target identification. In particular, the algorithm is derived from Bayes' rule as given by

$$P(h|Z) = \frac{P(h) \prod_{i=1}^N P_i(z_i|h)}{P(Z)} \quad (2.11)$$

where  $h$  is a realization of the hypothesis event in question within the space of all outcomes,  $H$ , and  $Z = \{z_1, \dots, z_N\}$  is the set of measurements by the  $N$  sensors. Each element of Bayes' rule plays a different role:

**A Priori Probability:** The *a priori* probability, often shortened to the prior, is the initial belief on the probability of  $h$ , denoted by  $P(h)$ .

**Likelihood:** The likelihood represents the probability of a measurement,  $z_i$ , being observed given  $h$ , and is denoted  $P(z_i|h)$ .

**A Posteriori Probability:** The *a posteriori* probability, or the posterior, is the result of Bayes' rule, and defines the final probability of  $h$  being correct given the prior information and any measurements. It is denoted  $P(h|Z)$ .

The denominator,  $P(Z)$ , is independent of whether or not  $h$  is true, and can therefore be considered a constant required to normalize the resulting probabilities. Note that each hypothesis is considered independently of the others (ie. determining whether  $h$  is true or false), such that  $\sum_{h \in H} P(h)$  does not necessarily sum to unity. It is assumed that each agent has an independent belief about the likelihood of  $h$  that needs to be shared with other agents according to Eq. 2.11.

In accordance with the terminology in [6], setting  $Q = \prod_{i=1}^N P_i(z_i|h)$  and defining  $\pi_i = P_i(z_i|h)$  as the *belief* of agent  $i$  and  $l_i = \log(\pi_i)$  as the *likelihood of the belief*, it

follows that

$$\begin{aligned}\log(Q) &= \log\left(\prod_{i=1}^N \pi_i\right) = \sum_{i=1}^N \log(\pi_i) \\ &= \sum_{i=1}^N l_i = N \sum_{i=1}^N \frac{l_i}{N}\end{aligned}$$

Thus, Olfati-Saber *et al.* suggest running the average consensus method in Eq. 2.2 on  $l_i$  in order to agree upon  $\log(Q)/N$ , such that each agent can then obtain  $Q$  by running an average consensus over a balanced network:

$$\begin{aligned}Q &= \exp\left\{N \sum_{i=1}^N \lim_{k \rightarrow \infty} l_i[k]\right\} \\ &= \exp\left\{N \sum_{i=1}^N \frac{l_i[0]}{N}\right\} = \exp\left\{\sum_{i=1}^N \log(\pi_i[0])\right\} \\ &= \prod_{i=1}^N \pi_i[0]\end{aligned}$$

Thus, BC can obtain its desired result under the same fairly standard network and communication assumptions as AC, but, through utilizing simple aspects of Bayesian probability theory, can achieve a more complex consensus result on beliefs over hypotheses. The result obtained with BC allows for some preliminary concept of uncertainty to be defined over a finite set of hypotheses, but is not directly applicable to continuous hypothesis spaces and, like average consensus, does not consider any uncertainty in the local beliefs themselves.

### 2.3.3 Kalman Consensus

The Kalman consensus method presented in [4] and modified in [5] is currently the primary method that explicitly considers the uncertainties in local estimates. The derivation is very similar to the sensor fusion problem of distributed Kalman filtering pioneered by Durrant-Whyte [34, 36, 37], and is based on the decentralization of Bayesian updates. Appendix A shows the Bayesian derivation of the Kalman filter

and associated Information filter, though only the results of the derivation will be repeated here.

It can be shown that the Kalman filter is the optimal Bayesian update scheme for the estimation of the mean of a normally distributed random variable with known second moment properties [46]. This discussion focuses on the discrete-time form of the filter, such that the second moment properties are captured by a covariance matrix, though similar discussions hold in the continuous time domain using an intensity matrix to represent second moment properties. In most contexts, the mean to be estimated represents the state of a process with known dynamics and affected by white, zero-mean Gaussian process noise with known covariance. Measurements are taken from a likelihood distribution<sup>3</sup> that is also normally distributed with the mean equal to the true state and covariance given by the covariance of the sensing noise. In this case, Bayes' rule is again invoked, though now in the generic form

$$p(\theta|z, \omega) = \frac{p(z|\theta, \omega)p(\theta|\omega)}{\int_{\Theta} p(z|\theta, \omega)p(\theta|\omega)d\theta} \quad (2.12)$$

where  $\theta$  is the estimate of the state,  $z$  is a measurement made from the sensing model, and  $\omega$  is any information that defines the prior distribution on  $\theta$ . Also, in Eq. 2.12 it is explicitly assumed that  $p(\cdot)$  denotes a probability distribution in discrete or continuous space as required by the circumstances. The Kalman filter utilizes the concept of *conjugacy* of distributions, whereby a prior distribution is called conjugate to a likelihood function if the prior and posterior are both of the same functional form (ie. same type of distribution). In the case of estimating the mean of a normal distribution, the conjugate prior defining the distribution on the mean is also Normally distributed and defined, itself, by a mean and variance. In the process of running a Kalman filter, it is this mean and variance that gets updated, redefining the posterior (and subsequent prior) distribution.

The key component to the Kalman consensus filter is the assumption that multiple

---

<sup>3</sup>The likelihood distribution is equivalent an extension of the likelihood probability defined in the belief consensus section, though now represents the likelihood of a parameter taking any value within in its domain given a measurement.

agents' uncertain knowledge can be treated as pseudo-measurements by the other agents, and can therefore be integrated through the same update equations as the Kalman filter itself. In the KC format, the state to be estimated,  $\xi^*$ , is assumed to have trivial dynamics and is only updated by some white, zero-mean Gaussian process noise,  $w$ , with covariance  $Q$ :

$$\xi^*[k+1] = \xi^*[k] + w[k]$$

The pseudo-measurement for agent  $i$  is assumed to be of the form

$$\begin{aligned} z_i[k] &= \begin{bmatrix} g_{i1}[k](\xi_i[k] + \eta_{i1}[k]) \\ \vdots \\ g_{iN}[k](\xi_N[k] + \eta_{iN}[k]) \end{bmatrix} \\ &= \begin{bmatrix} g_{i1}[k]I \\ \vdots \\ g_{iN}[k]I \end{bmatrix} \xi^*[k] + \begin{bmatrix} g_{i1}[k](\xi_i[k] - \xi^*[k] + \eta_{i1}[k]) \\ \vdots \\ g_{iN}[k](\xi_N[k] - \xi^*[k] + \eta_{iN}[k]) \end{bmatrix} \\ &= H\xi^*[k] + v_i[k] \end{aligned}$$

where  $g_{ij}[k]$  is the  $i, j^{\text{th}}$  entry of the connectivity graph and  $\eta_{ij}[k]$  is the communication noise between agents  $i$  and  $j$  which, together with the estimation error, define the pseudo-measurement noise,  $v_i[k]$ . It is shown in [4] that this definition leads to a consensus of the form

$$P_i[k+1] = \left[ (P_i[k] + Q[k])^{-1} + \sum_{j=1}^N g_{ij}[k](P_j[k] + \Omega_{ij}[k])^{-1} \right]^{-1} \quad (2.13)$$

$$\xi_i[k+1] = \xi_i[k] + P_i[k+1] \sum_{j=1}^N g_{ij}[k](P_j[k] + \Omega_{ij}[k])^{-1}(\xi_j[k] + \eta_{ij}[k] - \xi_i[k]) \quad (2.14)$$

where  $\Omega_{ij}[k]$  is the covariance in the communication noise,  $\eta_{ij}[k]$ . Alighanbari [5] shows that this form of the filter is actually susceptible to biases introduced through unbalanced networks, and developed an extension called the Modified Decentral-

ized Kalman Consensus (MDKC) algorithm. A particular extension was noting the information form of the consensus filter, obtained by setting  $Y_i[k] = P_i[k]^{-1}$  and  $y_i[k] = Y_i[k]\xi_i[k]$ , has the simple measurement update

$$Y_i[k+1] = \sum_{j=1}^N a_{ij} Y_j[k] \quad (2.15)$$

$$y_i[k+1] = \sum_{j=1}^N a_{ij} y_j[k] \quad (2.16)$$

where the network weights,  $a_{ij}[k]$ , are selected such that

$$a_{ij} = \begin{cases} \frac{1}{2} & \text{if } i = j \\ \frac{g_{ij}[k]}{2 \sum_{k=1, k \neq j}^N g_{kj}[k]} & \text{if } (j, i) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}$$

This results in the adjacency matrix  $D$  being column-stochastic<sup>4</sup>, and that there exists a consensus right eigenvector,  $\varsigma$ , such that  $\lim_{k \rightarrow \infty} D^k = \varsigma \mathbf{e}^T$ . This results in the final steady-state consensus value derived as

$$\begin{aligned} \lim_{k \rightarrow \infty} \xi[k] &= \lim_{k \rightarrow \infty} (D^k Y[0])^\dagger \odot (D^k y[0]) \\ &= (\varsigma \mathbf{e}^T Y[0])^\dagger \odot (\varsigma \mathbf{e}^T y[0]) \\ &= \mathbf{e} \left\{ \sum_{i=1}^N Y_i[0] \right\}^{-1} \left\{ \sum_{i=1}^N y_i[0] \right\} \\ &= \mathbf{e} \left\{ \sum_{i=1}^N P_i^{-1}[0] \right\}^{-1} \left\{ \sum_{i=1}^N P_i^{-1}[0] \xi_i[0] \right\} \end{aligned}$$

which is equivalent to the centralized Bayesian consensus value as defined by the Kalman updates (shown in Appendix A).

The MDKC algorithm is guaranteed to converge to the true Bayesian centralized estimate of the mean over known, strongly connected networks and assuming each

---

<sup>4</sup>Note the difference between this result and the row-stochastic result in Eq. 2.5.

agent's uncertainty is modeled using a normal distribution. However, the MDKC algorithm has two primary deficiencies: first, the consensus value is only properly biased if the agents' knowledge is normally distributed. If agents are using other distributions to model their estimate then the use of first- and second-moment information in a KC algorithm may be invalid, and the consensus result may not necessarily be any more accurate than doing parameter consensus alone (recall Figures 1-1 and 1-2 in Chapter 1). Second, the filter converges to the proper unbiased mean, but does not converge to the unbiased covariance:

$$\lim_{k \rightarrow \infty} P_i[k] = \lim_{k \rightarrow \infty} Y_i^{-1}[k] = (\varsigma_i \mathbf{e}^T Y[0])^{-1} \neq (\mathbf{e}^T Y[0])^{-1}$$

The true covariance can be deduced using some algebra, though the algebra can only be performed once consensus has been reached. Thus, the resulting covariance trajectory converges to the wrong result, only to be compensated for after achieving some desired degree of convergence. This indicates that, if a process is being estimated concurrently, the agent will have significantly biased estimates through the entire consensus transient, which may impact the resulting long-term estimation.

## 2.4 Summary

This chapter has introduced three of the primary consensus algorithms from the current literature, two of which attempt to, in different manners, deal with forms of uncertainty in the value of a consensus parameter. Average consensus was shown to provide a robust method to achieve consensus over a network when there are no uncertainties associated with the agents' initial knowledge. Belief consensus utilizes some average consensus results to converge on a product of initial beliefs as motivated by Bayesian principles in order to estimate the most likely hypothesis or classification of an event or object. This form of consensus permits a measure of uncertainty on the value of a parameter over a fixed, discrete set of possibilities, but does not consider more general, continuous distributions, nor did it allow for uncertainties



in the local beliefs. Kalman consensus methods, on the other hand, do explicitly take into account the uncertainty in agents local estimates in achieving consensus. Further, they are able converge to the centralized, Bayesian estimate of the mean of a normal distribution with known covariance over any known, static, strongly connected network. However, it is not theoretically applicable to any uncertainties described by non-normal distributions, or, equivalently, it is not applicable when attempting to estimate distributions with non-normal conjugate priors.

Thus, this thesis develops a consensus method that allows for a broad range of local uncertainties to be taken into account in a rigorous Bayesian sense. This will be similar to both the belief consensus and Kalman consensus methods insofar as it is required to aggregate agents' information in a principled and well-defined manner. However, it will differ from the existing methods by addressing the question of uncertainties in initial beliefs that is lacking in BC, and will extend similarly exact methods as the KC algorithm to alternate distributions.



# Chapter 3

## Bayesian Hyperparameter

### Consensus

This chapter introduces the proposed Bayesian hyperparameter consensus method in response to the inability of many current consensus algorithms to properly account for generic local uncertainties in the agreement protocol as motivated in Chapters 1 and 2. With the possible exception of normally distributed local estimates, which have been examined through use of Kalman consensus filters, uncertainties characterized by many common probability distributions are not currently considered in the consensus community. These shortcomings will be addressed through the development of a consensus algorithm utilizing well-defined fundamentals of probability theory to guarantee convergence to the centralized Bayesian estimate of a distribution in the presence of parameterized initial uncertainties.

To achieve the desired goal, the proposed algorithm expands upon principles from both the consensus and data fusion community. In particular, the communication protocol will be adapted from existing average consensus methods, while the information aggregation will be largely based on Bayesian inference approaches utilized in data fusion. This approach will enable the consensus algorithm to reach an unbiased agreement on a broader range of uncertain parameters, while the key contribution to the data fusion community is the use of consensus methods to partially mitigate the requirement for complex channel filters that are required to avoid double-counting of

information (eg. [34]). Further, the use of generalized consensus algorithms over large, complex communication networks allows for a degree of scalability that is unobtainable with many current fusion architectures. Xiao *et al.* [39] successfully demonstrated an application of average consensus to the distributed sensor fusion problem, though they assumed a Kalman filter context and an undirected network. This approach will be extended to a more generic class of distributions with provable convergence to the desired centralized Bayesian result. It will also be shown that the proposed method converges to the desired parameter estimate, as well as an unbiased representation of the uncertainty.

The chapter starts with the derivation of the desired centralized Bayesian value and the primary assumptions of the proposed method. The centralized result will then be used to motivate a new *hyperparameter consensus method* that is guaranteed to converge to the desired values in the static case as well as when each agent is changing its information locally due to measurements or other inputs. It is shown that, in the case of independent measurements, the resulting consensus converges to the proper, changing centralized estimate, such that agents can simultaneously measure and agree upon an uncertain parameter of interest. Finally, the static and dynamic approach will be demonstrated on two primary uncertainty distributions of interest: the gamma and Dirichlet.

### 3.1 Derivation

This first section outlines the format of the problem, some of the primary assumptions, and describes how to derive the centralized result. It then formalizes the hyperparameter consensus method and provides theoretical guarantees on the convergence result in both the case of both static and dynamic local estimates.

The primary problem of interest is effectively that of agreement on a probability distribution. While this seems to be a new and challenging concept, it is actually the principle behind Kalman filtering and the Kalman consensus algorithm, though tailored to the normal distribution. By understanding the Bayesian principles behind

Kalman filtering, and the pseudo-measurement extension to the Kalman consensus algorithm, many insights can be developed for the general case. The following brief discussion is intended to illustrate some of the following concepts in the context of a well known algorithm, while also motivating the derivation of the hyperparameter consensus method.

### 3.1.1 The Bayesian Kalman Filter

When an agent runs a Kalman filter, it is executing an iterative Bayesian inference scheme to estimate the mean,  $\mu$ , of a normal distribution with known covariance. The result of this approach is an estimate of the mean,  $\hat{x}$ , and error covariance of that estimate,  $P$ . These variables actually define a conjugate normal distribution over the value of the mean,  $\mu$ ,

$$p(\mu|\hat{x}, P) = \mathcal{N}(\hat{x}, P) \quad (3.1)$$

such that the best estimate of the mean,  $\mu$ , is equivalent to the mean of the distribution,  $\hat{x}$  (see Appendix A for more information).

Before continuing, it is important to note that there are many metrics to define what the “best” estimate of the parameter is given a distribution over its domain, all of which can be defined as minimizing a loss function or maximizing a utility. If a constant utility function is used, the resulting best estimate is the maximum *a posteriori* (MAP) estimate:

$$\hat{\mu}_{MAP} = \arg \max_{\mu} p(\mu|\hat{x}, P) \quad (3.2)$$

This estimate doesn’t take into account any higher-order properties of the distribution, and so the mean square error (MSE) loss function is often used in its place:

$$E[(\hat{\mu} - \mu)^2] \quad (3.3)$$

Using this loss function (which is equivalent to the posterior covariance), the resulting minimum mean square error (MMSE) estimate is simply the mean of the posterior

distribution:

$$\hat{\mu}_{MMSE} = \int \mu p(\mu|\hat{x}, P) d\mu = \hat{x} \quad (3.4)$$

Since the mode, mean, and maximum of the normal distribution all occur at the same location, most metrics will result in an estimate for  $\mu$  equal to the mean of the posterior,  $\hat{x}$ . Though there are, of course, infinite loss and utility functions, this thesis will primarily use the MMSE estimate due to its representative nature and frequent use in the Bayesian inference community.

The normal distribution in Eq. 3.1 represents either the prior (before measurement) or posterior (after measurement) distribution on the unknown mean value,  $\mu$ . The simple, closed form updates of the Kalman filter are possible because of two properties: the normal distribution is uniquely defined by a set of parameters (the mean and covariance), and the normal distribution is conjugate to a normal likelihood function. Thus, taking a sample from the measurement model, which is given as a normal distribution with sensor noise covariance,  $R$ , and defines the likelihood function, and applying that measurement to Bayes' rule in Eq. 2.12 with a normal prior distribution produces another normal distribution as the posterior with parameters updated through the traditional Kalman filter equations.

By examining the Kalman filter, a number of observations can be made. First, for parameterized distributions such as the normal distribution, agreement on the distribution and agreement on the distribution's parameters are equivalent. Second, though the parameter itself may be of primary interest, the form of the sampled likelihood and uncertainty distributions are important. If a non-normal prior was selected with the normal sensing model, then the resulting update would not be available in as succinct a representation as is available through use of the normal conjugate prior. Third, the primary parameters that are saved and updated in the Kalman filter,  $\hat{x}$  and  $P$ , are actually the parameters of the conjugate distribution, hereafter termed the *hyperparameters* [41, 47] to differentiate them from the parameters of the likelihood function that is being estimated (in this case,  $\mu$ ). These hyperparameters define the uncertainty on the parameter and are updated in the Bayesian sense through simple

closed form equations. Finally, though  $P$  and  $\hat{x}$  have nonlinear updates, a nonlinear transformation to the information filter hyperparameters,  $Y$  and  $y$ , allows for simple additive updates.

### 3.1.2 The Centralized Bayesian Estimate

Consider the task of consensus on a probability distribution  $f_{X|\Theta}(x|\theta)$ , where  $X$  is a random variable and the distribution is parameterized by a (possibly multivariate) parameter,  $\theta$ . Let  $f_{X|\Theta}$  have a conjugate prior distribution,  $f_{\Theta|\Omega}(\theta|\omega)$ , which defines the distribution on the parameter  $\theta$  based on hyperparameters,  $\omega$ . Finally, define a *non-informative* prior [48]<sup>1</sup>,  $f_{\Theta}(\theta)$ , and the likelihood belief as

$$\pi(\omega|\theta) = \frac{f_{\Theta|\Omega}(\theta|\omega)}{f_{\Theta}(\theta)} \quad (3.5)$$

It is assumed that the agents have agreed a priori on the form of the distributions in question, such that the functional form of the likelihoods and priors are consistent across the network (ie. all normal or gamma etc.), and that their initial estimates are independent. Let agent  $i$ 's information be denoted by  $(\cdot)_i$ , both in terms of parameter estimates and probability distributions, and also let  $p$  denote a generic intermediate probability distribution as defined by the context. The centralized estimate can then be found using Bayes rule as

$$\begin{aligned} f_{\Theta|\Omega}(\theta|\omega_1, \dots, \omega_N) &\propto p(\theta, \omega_1, \dots, \omega_N) = \pi(\omega_1|\theta, \omega_2, \dots, \omega_N)p(\theta, \omega_2, \dots, \omega_N) \\ &= \prod_i^N \pi(\omega_i|\theta) f_{\Theta}(\theta) \\ &\propto \prod_i^N \left( \frac{f_{\Theta|\Omega}(\theta|\omega_i)}{f_{\Theta}(\theta)} \right) f_{\Theta}(\theta) \end{aligned} \quad (3.6)$$

---

<sup>1</sup>A non-informative prior is a prior that attempts to provide no initial knowledge on the parameter [47], such that the resulting 'best' posterior estimate (MAP, MMSE, etc) is roughly equivalent to the maximum likelihood estimate (MLE) given a measurement,  $z$ , where  $\hat{\theta}_{MLE} = \arg \max_{\theta} f_{X|\Theta}(z|\theta)$ . These priors are difficult to define, and will be addressed further in Section 3.2.1.

where the product term arises due to the independence of the agents' information (and, therefore, the independence of their hyperparameters). Equation 3.6 is similar in form to Eq. 2.11 from belief consensus, though now the likelihood belief  $\pi_i = \pi(\omega_i|\theta)$  is a function of the parameters and hyperparameters rather than a scalar. This complicates the generic form of centralized estimate, but it is possible to use the conjugacy property of the distributions to simplify this result.

First, consider Bayes' rule applied to a single agent with a prior,  $f_{\Theta}(\theta)$ . The update for a measurement  $Z = z$  is

$$f_{\Theta|Z}(\theta|z) \propto f_{X|\Theta}(z|\theta)f_{\Theta}(\theta)$$

which can be rearranged to get

$$f_{X|\Theta}(z|\theta) \propto \frac{f_{\Theta|Z}(\theta|z)}{f_{\Theta}(\theta)} = \pi(z|\theta) \quad (3.7)$$

Thus, noting the functional equivalence between Eq. 3.7 and 3.5,  $\pi(\omega|\theta)$  can be interpreted as representing a pseudo-measurement that would have been applied to the non-informative prior  $f_{\Theta}(\theta)$  to get  $f_{\Theta|\Omega}(\theta|\omega)$  as the resulting posterior. Of primary importance is the form of the hyperparameter update for a given likelihood/conjugate prior pair. Fortunately, most conjugate updates have remarkably simple additive updates of the form

$$\omega \leftarrow \omega + h(z) \quad (3.8)$$

where  $h(z)$  is a generic operator that obtains the relevant quantities from the measurement and measurement model as are required for the hyperparameter update (see Section 3.2.1 for an illustrative example). Some distributions, such as in the Kalman filter case, may require a transformation of hyperparameters in order achieve this result, but many are naturally of this form.

At this point, it is convenient to explain the use of the non-informative prior in the consensus framework: What this implies is that, given each agent's initial information is independent of the other agents', the entirety of the agent's belief needs to be



communicated to ensure consensus. Since the hyperparameters uniquely define the agent’s information, the complete set of hyperparameters should be shared, which can easily be achieved by assuming the proper non-informative *consensus prior*. The selection of non-informative priors for specific cases will be discussed in the examples in Section 3.2. In the current generic case, the non-informative prior can be considered roughly equivalent to the conjugate prior with the hyperparameters  $\omega$  set to 0 or a similar *null* value, such that the update in Eq. 3.8 can be considered as the equality

$$\omega_i = h(z_i) \tag{3.9}$$

for the consensus problem, where  $z_i$  is the corresponding pseudo-measurement.

Finally, Eq. 3.6 can be evaluated by noting that the centralized estimate is obtained using the same consensus prior and aggregating  $N$  pseudo-measurements, defined by the likelihood functions  $\pi$ , to achieve a *consensus posterior*. Thus, the same closed-form updates hold for the consensus problem as held in the inference problem, and the centralized result can be found as

$$\begin{aligned} f_{\Theta|\Omega}(\theta|\omega_1, \dots, \omega_N) &= f_{\Theta|\Omega} \left( \theta \left| \omega = \sum_{i=1}^N h(z_i) \right. \right) \\ &= f_{\Theta|\Omega} \left( \theta \left| \omega = \sum_{i=1}^N \omega_i \right. \right) \end{aligned} \tag{3.10}$$

Therefore, the centralized Bayesian estimate is defined as the consensus posterior with hyperparameters equal to the summation of each agent’s local hyperparameters.

### 3.1.3 The Hyperparameter Consensus Method

With the desired centralized estimate properly defined, it is now possible to develop a consensus protocol that will achieve this result. While belief consensus [6] focuses on algorithms derived from Eq. 3.6, this approach is complicated here by the fact that  $\pi_i$  is no longer a scalar value as is assumed in the literature, but rather a function of the parameters and hyperparameters of the distributions. Currently, the consensus

community has no explicit methods for consensus on generic functions, so it is not immediately clear how to come to consensus on these  $\pi_i$ 's. However, if the form of the function is known and parameterized, then consensus on the parameters of the function implicitly aligns the functions themselves.

Thus, instead of considering Eq. 3.6, it is beneficial to look to Eq. 3.10 to define the required consensus protocol. In particular, the latter equation states that the centralized parameter estimate is distributed according to the conjugate distribution which is defined by a set of hyperparameters. Therefore, since all agents have agreed on the distributions to use ahead of time (this assumption was stated in the previous section), then consensus on the hyperparameters will lead to consensus on the conjugate distributions themselves. Further, if the conjugate distributions are in agreement, then the agents will also be in agreement on the best estimate of the parameters and, ultimately, come to agreement on the likelihood distribution that the parameters define and any corresponding random variables that could be obtained therefrom.

The question now becomes one of how to agree properly on the hyperparameters, which, for simplicity, will be assumed scalar for the following discussion. Eq. 3.10 shows that the centralized estimate can be found using the pure summation of each agent's local hyperparameters. Therefore, if the same result can be obtained through a consensus protocol then the centralized Bayesian estimate is obtained locally by each agent.

This goal of a sum-consensus on the hyperparameters can be achieved by recalling previous average consensus results from Section 2.3.1, where a sum-consensus on the hyperparameter can be achieved by running the update

$$\omega_i[k+1] = \sum_{j=1}^N a_{ij} \omega_j[k] \quad (3.11)$$

with

$$\omega[0] = \text{diag}(\nu^\dagger) \omega[0^-] \quad (3.12)$$

where  $\omega[0^-]$  and  $\omega[0]$  are the un-weighted and weighted initial hyperparameters, respectively, and the adjacency matrix is composed of entries,  $a_{ij}$ , defined as in Eq. 2.4 and 2.3, with corresponding consensus eigenvalue,  $\nu$ . As was shown in Section 2.3.1, the protocol defined by Eq. 3.11 and 3.12 will converge to the sum of the initial, un-weighted hyperparameters. The primary results is stated in the following theorem:

**Theorem 3.1.1** (Convergence on Distributions). *A group of  $N$  agents can come to asymptotic agreement on a likelihood distribution,  $f_{X|\Theta}(x|\theta)$ , that is parameterized uniquely by  $\theta$ , through use of the derived hyperparameter consensus algorithm on the hyperparameters,  $\omega$ , and under the following properties:*

1. *the connectivity graph  $\mathcal{G}$  is time-invariant, strongly connected and known<sup>2</sup>,*
2. *the associated adjacency matrix,  $D$ , has entries  $a_{ij}$  as defined as in Eq. 2.4 and 2.3, and consensus eigenvalue  $\nu$ ,*
3. *the agents' maintain uncertain local estimates of  $\theta$  through  $f_{\Theta|\Omega}$ , the conjugate distribution to  $f_{X|\Theta}$ ,*
4. *each agent's initial information is independent of all other agent's information, and*
5. *the agents have decided a priori on the form of the distribution to agree upon<sup>3</sup>.*

*Proof of Theorem 3.1.1.* Note, first, that  $f_{X|\Theta}(x|\theta)$  is uniquely defined by the parameters,  $\theta$ . Thus, if each agent agrees on estimates of  $\theta$ , then they will implicitly agree upon the distribution itself. Second,  $\theta$  is distributed according to the conjugate distribution,  $f_{\Theta|\Omega}(\theta|\omega)$ , such that agreement on the hyperparameters,  $\omega$ , will lead to a similar convergence on the conjugate distribution. Finally, Eq. 3.10 gives an expression for the centralized hyperparameter value (achieved assuming properties 3-5)

---

<sup>2</sup>The known network property is sufficient, but not always necessary. The goal is to know the resulting value of  $\nu$ , which may be known without knowing the exact network topology (eg. undirected networks are inherently balanced  $\rightarrow \nu = \mathbf{e}/N$ ).

<sup>3</sup>This is not as restrictive as it might seem since this is always the case with any Kalman-derived method, though confined to the normal distribution. Often, the selection of distribution is derived automatically from the problem at hand (see Section 3.2).

that, if obtained, will implicitly achieve agreement to the centralized conjugate distribution, and, subsequently, the parameter and likelihood distribution. Therefore, it is sufficient to prove convergence to the sum of initial hyperparameter values as in Eq. 3.10.

Property 1 implies the existence of the strictly positive consensus eigenvalue  $\nu$  [26], such that the weights in 3.12 are well defined. This property, combined with property 2, also ensures existence of a steady-state value if each agent runs the consensus update in Eq. 3.11 [3], which is found to be:

$$\begin{aligned}
\lim_{k \rightarrow \infty} \omega[k] &= \lim_{k \rightarrow \infty} D^k \omega[0] \\
&= \mathbf{e} \underbrace{\nu^T \text{diag}(\nu^\dagger)}_{\mathbf{e}^T} \omega[0^-] \\
&= \mathbf{e} \mathbf{e}^T \omega[0^-] \\
&= \mathbf{e} \left( \sum_{i=1}^N \omega_i[0^-] \right) \tag{3.13}
\end{aligned}$$

Noting that the  $\omega_i$  terms in Eq. 3.10 are equivalent to the  $\omega_i[0^-]$  terms in Eq. 3.14, the desired centralized hyperparameter estimate has been obtained by each agent.

By definition, these values uniquely define the conjugate distribution over the likelihood parameter  $\theta$ , such that the arrived at agreement on the hyperparameters also implies agreement on these conjugate distributions. Further, since all agents have converged to the centralized distribution on  $\theta$  given in Eq. 3.10, and are all assumed to be using the same MMSE loss function, the agents have also implicitly agreed on the centralized Bayesian estimate of the parameter. Finally, the parameter itself uniquely defines the likelihood distribution  $f_{X|\Theta}(x|\theta)$ , which implies that all agents have also achieved agreement on the likelihood distribution, as stated in the theorem.  $\square$

Thus, it has been shown that the conjugacy property of certain distributions can be exploited to perform a distributed Bayesian aggregation of local information using simple additive updates of the hyperparameters. Further, by formulating local uncertainties according to a distribution that is conjugate to a likelihood function, the

pseudo-measurement distribution  $\pi(\omega_i|\theta)$  in Eq. 3.5 is guaranteed to be of the form of the likelihood function and, therefore, ensures the required conjugacy property and the existence of the simple hyperparameter updates.

Finally, this approach allows agents to agree upon the centralized posterior distribution over the defining parameters of a broad range of parameterized likelihood distributions. As such, not only does this method address the problem of uncertain local estimates of a not necessarily Normally distributed parameter, but if that parameter uniquely defines a subsequent distribution then the agents can also agree upon that distribution and the generation of any random variables associated with it.

### Handling Shared Information

So far, we have assumed that there is no globally shared information between agents. If this is not the case, such as if agents have previously come to agreement, made local changes, and must agree again, then the algorithm described thus far would incorrectly count the shared information from the first consensus as new, independent information from each agent. To prevent this, two approaches are available:

1. Only come to convergence on the *new* information since the last consensus. This requires subtracting the hyperparameters corresponding to the shared information from the current local counts, running consensus on the difference, and adding the result of the consensus back to the global information once convergence has been reached to reconstruct the centralized hyperparameter estimate.
2. Scale the local information such that the result is an effective *average* consensus on the shared information, and a *sum* consensus on the new measurements. This does not require any explicit “reconstruction” step, but does also require knowledge of the shared information.

Both approaches require knowledge of the mutual information between agents, somewhat akin to a channel filter in data fusion approaches. Fortunately, in the context

considered here, any shared knowledge after a consensus is *globally* shared knowledge, and doesn't require explicitly maintaining a channel filter between each connected agent. This key difference allows for each agent to simply record a single copy of the hyperparameters that resulted from the most recent consensus and save them as the global information, rather than explicitly keeping track of the shared information between it and each of the other agents as is required with conventional data fusion methods.

At this point, it is necessary to introduce a second time-scale,  $(t)$ , that specifies epochs of consensus or measurements, where a consensus epoch is the complete execution from initial weighting to convergence of the consensus algorithm, and a measurement epoch is defined as a measurement update obtained outside of any consensus epoch. The epoch time is incremented after each epoch, such that a consensus on the hyperparameters initiated at time  $(t)$  leads to a converged hyperparameter estimate at time  $(t + 1)$ . Further, any shared hyperparameters at time  $(t)$  will be denoted as  $\omega(t^-)$ , such that the new information for an agent  $i$  is

$$\Delta\omega_i(t) = \omega_i(t) - \omega(t^-)$$

Returning to the question of how to address this shared information, the first approach can be achieved by running the consensus in Eq. 3.11 and 3.12 on the difference between the local hyperparameters and any shared information,  $\Delta\omega_i(t)$ . To obtain the parameter estimate, the current hyperparameter estimate then needs to be added to the shared information

$$\omega_i(t + 1) = \omega(t^-) + \lim_{k \rightarrow \infty} \Delta\omega_i[k] = \omega(t^-) + \sum_{j=1}^N \Delta\omega_j[k]$$

where  $\Delta\omega_i[k]$  is the local estimate of the consensus value at iteration  $k$ , and is initialized as

$$\Delta\omega[0] = \text{diag}(\nu^\dagger)\Delta\omega[0^-] = \text{diag}(\nu^\dagger)\Delta\omega_i(t)$$

The second proposed method is equivalent to the first but does not require the explicit re-addition of the shared information to the current consensus estimate. Instead, the weighted initial conditions for the consensus as derived in Eq. 3.12 are replaced by

$$\omega_i[0] = \omega(t^-) + \frac{\Delta\omega_i(t)}{\nu_i}$$

such that

$$\begin{aligned} \lim_{k \rightarrow \infty} \omega[k] &= \lim_{k \rightarrow \infty} D^k \omega[0] \\ &= \lim_{k \rightarrow \infty} D^k (\mathbf{e}\omega(t^-) + \text{diag}(\nu^\dagger)\Delta\omega(t)) \\ &= \mathbf{e} \left( \underbrace{\nu^T \mathbf{e}}_1 \omega(t^-) + \underbrace{\nu^T \text{diag}(\nu^\dagger)}_{\mathbf{e}^T} \Delta\omega(t) \right) \\ &= \mathbf{e} (\omega(t^-) + \mathbf{e}^T \Delta\omega(t)) \\ &= \mathbf{e} \left( \omega(t^-) + \sum_{i=1}^N \Delta\omega_i(t) \right) \end{aligned} \tag{3.14}$$

Both approaches are equivalent and it is a matter of preference as to which one to use for a given situation. If an estimate of the parameter is likely to be required during the execution of the consensus, the second method may be preferable since the hyperparameters are directly accessible as the most recent local consensus value. The complete method for the second approach is outlined in Algorithm 1.

## Hyperparameter Consensus with Measurements

In addition to convergence on static local information, the hyperparameter consensus method allows for the local modification of information through measurements of a static process<sup>4</sup> while concurrently running the consensus algorithm. Once a local modification has been made and incorporated into the agent's consensus framework, the consensus scheme will automatically begin to converge to the new, modified centralized estimate.

---

<sup>4</sup>Though not considered in this thesis, some thoughts on the problem derived from estimating a time-varying process are given in the discussion section of this chapter.

---

**Algorithm 1** Hyperparameter consensus method with globally shared information

---

- 1: Choose a convergence criterion
- 2: Initialize hyperparameters:  $\omega_i[0] \leftarrow \omega(t^-) + \frac{\omega_i(t^-) - \omega(t^-)}{\nu^i}$
- 3: Initialize iteration counter:  $k \leftarrow 0$
- 4: **while** Convergence criterion not satisfied **do**
- 5:   Update hyperparameters:

$$\omega_i[k+1] = \sum_{j=1}^N a_{ij} \omega_j[k]$$

- 6:   Find the MMSE estimate:

$$\hat{\theta}_i[k+1] = \int_{-\infty}^{\infty} \theta f_{\Theta|\Omega}(\theta|\omega_i[k+1]) d\theta$$

- 7:   Update counter:  $k \leftarrow k+1$
  - 8:   Evaluate convergence criterion
  - 9: **end while**
  - 10: Update local parameter estimate:  $\hat{\theta}_i \leftarrow \hat{\theta}_i[k]$
- 

**Theorem 3.1.2** (Convergence with Measurements). *For  $N$  agents running the consensus algorithm defined by Equations 3.11 and 3.12 with the properties in Theorem 3.1.1 holding true, then if the agents take independent measurements  $z_i[k]$  at times  $k = 0, \dots, K$ , with  $0 \leq K < \infty$ , with  $\mathbf{z}[k] = [z_1[k], \dots, z_N[k]]^T$ , then the agents will converge locally to the new centralized result.*

Before proceeding with the proof, the properties of the centralized estimate when measurements are involved is examined in the following proposition:

**Proposition 3.1.3** (Centralized Estimate with Measurements). *When a series of independent measurements is made by each agent,*

$$z_1[0], \dots, z_N[0], \dots, z_1[K], \dots, z_N[K]$$

for  $K \in [0, \infty)$ , the centralized result at  $K$  is given by:

$$\omega_{cent}[K] \leftarrow \omega_{cent}[0^-] + \sum_{i=1}^N \sum_{j=1}^K h(z_i[j]) \tag{3.15}$$



where  $\omega_{cent}[0^-]$  is the centralized result prior to any measurements, and is equal to the sum of the initial initial local hyperparameters,  $\sum_{i=1}^N \omega_i[0^-]$ .

*Proof of Proposition 3.1.3.* When a new measurement,  $z$ , is taken, the centralized result should be updated as if the measurement were applied to it directly. This implies a centralized hyperparameter update of the form:

$$\omega_{cent} \leftarrow \omega_{cent} + h(z) \tag{3.16}$$

With repeated independent measurements,

$$z_1[0], \dots, z_N[0], \dots, z_1[K], \dots, z_N[K]$$

the update in Eq. 3.16 becomes recursive, leading to:

$$\omega_{cent}[K] \leftarrow \omega_{cent}[0^-] + \sum_{i=1}^N \sum_{j=1}^K h(z_i[j])$$

where  $\omega_{cent}[0^-] = \sum_{i=1}^N \omega_i[0^-]$ . Further, if no subsequent measurements are taken, then for  $k \geq K$  it holds that  $\omega_{cent}[k] = \omega_{cent}[K]$ . An alternate view is that, if consensus were initiated after all the measurements were taken, then the post-measurement centralized estimate should be the same as if the measurements were made during consensus, and would then be equivalent the sum of all the local hyperparameters after all measurements:

$$\omega_{cent}[k] \leftarrow \sum_{i=1}^N \left( \omega_i[0^-] + \sum_{j=1}^k h(z_i[j]) \right)$$

which leads immediately to the same result. □

With this understanding of the centralized estimate, it is now possible to complete the proof of Theorem 3.1.2:

*Proof of Theorem 3.1.2.* Consider an agent  $i$  that makes a measurement  $z_i[\kappa]$  at some time  $\kappa$ , and that this measurement is incorporated into that agent's local information

as

$$\omega_i[\kappa] \leftarrow \omega_i[\kappa] + \frac{h(z_i[\kappa])}{\nu_i} \quad (3.17)$$

If all the agents make independent measurements at time  $\kappa$ , and letting  $\mathbf{z}[\kappa] = [z_1[\kappa], \dots, z_N[\kappa]]^T$ , then the resulting information update across all agents will be

$$\omega[\kappa + 1] = D (\omega[\kappa] + \text{diag}(\nu^\dagger)h(\mathbf{z}[\kappa]))$$

Therefore, letting measurements be taken for all  $\kappa$  such that  $0 \leq \kappa \leq K < \infty$ , where  $z_i[\kappa] = \emptyset$  if no measurement is made by agent  $i$  at time  $\kappa$ , then the state of information across the network at time  $k \leq K$  is:

$$\begin{aligned} \omega[k + 1] &= D (\omega[k] + \text{diag}(\nu^\dagger)h(\mathbf{z}[k])) \\ &= D (D (\omega[k - 1] + \text{diag}(\nu^\dagger)h(\mathbf{z}[k - 1])) + \text{diag}(\nu^\dagger)h(\mathbf{z}[k])) \\ &= D^k \omega[0] + \sum_{i=0}^k D^{k-i} \text{diag}(\nu^\dagger)h(\mathbf{z}[i]) \end{aligned} \quad (3.18)$$

Taking the limit of infinite communication, the resulting consensus value is then:

$$\begin{aligned} \lim_{k \rightarrow \infty} \omega[k + 1] &= \lim_{k \rightarrow \infty} \left( D^k \omega[0] + D^{k-K} \sum_{i=0}^K D^{K-i} \text{diag}(\nu^\dagger)h(\mathbf{z}[i]) \right) \\ &= \mathbf{e} \nu^T \omega[0] + \mathbf{e}^T \nu^T \sum_{i=0}^K D^{K-i} \text{diag}(\nu^\dagger)h(\mathbf{z}[i]) \\ &= \mathbf{e} \left( \underbrace{\nu^T \text{diag}(\nu^\dagger)}_{\mathbf{e}^T} \omega[0^-] + \sum_{i=0}^K \underbrace{\nu^T D^{K-i}}_{\nu^T} \text{diag}(\nu^\dagger)h(\mathbf{z}[i]) \right) \\ &= \mathbf{e} \left( \mathbf{e}^T \omega[0^-] + \sum_{i=0}^K \underbrace{\nu^T \text{diag}(\nu^\dagger)}_{\mathbf{e}^T} h(\mathbf{z}[i]) \right) \\ &= \mathbf{e} \left( \mathbf{e}^T \omega[0^-] + \sum_{i=0}^K \mathbf{e}^T h(\mathbf{z}[i]) \right) \\ &= \mathbf{e} \left( \sum_{i=1}^N \left( \omega_i[0^-] + \sum_{j=0}^K h(z_i[j]) \right) \right) \end{aligned} \quad (3.19)$$

---

**Algorithm 2** Hyperparameter consensus method with local measurement updates

---

- 1: Choose a convergence criterion
- 2: Initialize hyperparameters:  $\omega_i[0] \leftarrow \omega(t^-) + \frac{\omega_i(t) - \omega(t^-)}{\nu^i}$
- 3: Initialize iteration counter:  $k \leftarrow 0$
- 4: **while** Convergence criterion not satisfied **do**
- 5:   **if** Measurement received by agent  $i$  **then**
- 6:     Update local hyperparameters:  $\omega_i[k] \leftarrow \omega_i[k] + \frac{h(z_i[k])}{\nu^i}$
- 7:   **end if**
- 8:   Update hyperparameters:

$$\omega_i[k+1] = \sum_{j=1}^N \omega_j[k]$$

- 9:   Find the MMSE estimate:

$$\hat{\theta}_i[k+1] = \int_{-\infty}^{\infty} \theta f_{\Theta|\Omega}(\theta|\omega_i[k+1])d\theta$$

- 10:   Update counter:  $k \leftarrow k + 1$
  - 11:   Evaluate convergence criterion
  - 12: **end while**
  - 13: Update local hyperparameters:  $\omega_i(t+1) \leftarrow \omega_i[k]$
  - 14: Update local parameter estimate:  $\hat{\theta}_i \leftarrow \hat{\theta}_i[k]$
  - 15: New local distribution is  $f_{\Theta|\Omega}(\theta|\omega_i(t+1))$
  - 16: Update the time step  $t \leftarrow t + 1$
- 

Noting that  $\omega_{cent}[0^-]$  in Eq. 3.15 is equivalent to  $\sum_{i=1}^N \omega_i[0^-]$ , it follows directly that Eq. 3.19 is exactly the centralized estimate with all measurements considered.  $\square$

As with the static case, mutual information can also be accounted for in the consensus on dynamic local estimates through the same augmentations to the initial conditions. Algorithm 2 shows the algorithm with shared initial information and measurements occurring concurrently with the consensus.

## 3.2 Illustrative Examples

This section introduces two formulations of interest that will be used to highlight the application of the hyperparameter consensus method as well as demonstrate its results. While the consensus method is applicable to many distributions, two particular

example cases will be used to illustrate the use of the method. First is a scalar example of consensus on the value of an arrival rate using the gamma prior, and second is the application of hyperparameter consensus to a multivariate probability vector using the Dirichlet prior. Though discussion here will be limited to these two distributions, Table 3.1 highlights a selection of the other distributions that can be agreed upon with the hyperparameter approach, as well as their parameters, hyperparameters, and conjugate and consensus priors.

### 3.2.1 The Gamma Prior

Though the gamma distribution is conjugate to many different distribution's parameters [47], such as the variance of normal distribution with known mean, the shape parameter of a Pareto distribution, and the gamma's own rate parameter, this section focuses on the gamma prior as it pertains to the estimation of the arrival rate parameter,  $\lambda$ , associated with the Poisson and exponential distributions. This is an important parameter in terms of reliability analysis (how often equipment breaks down), scheduling (how many cars to expect at a toll checkpoint), and physics (distribution of arrivals of particles at a detector).

When considering sequential events in time, sequences are often modeled as Poisson Point Processes. This assumption carries with it two standard ways of measuring the data: number of arrivals,  $k$ , in a given period,  $T > 0$ ; and the inter-arrival time,  $t$ , between subsequent arrivals. In the first case, the random variable  $k$  is defined by a Poisson distribution, shown in Equation 3.20, while, in the second case,  $t$  is defined by an exponential distribution, shown in Equation 3.21. Both distributions are characterized by the arrival rate  $\lambda$ , but the Poisson distribution is further described by the period,  $T$ .

$$f_P(k|\lambda, T) = \frac{(\lambda T)^k e^{-\lambda T}}{k!} \quad k = 0, 1, 2, \dots \quad (3.20)$$

$$f_E(t|\lambda) = \lambda e^{-\lambda t} \quad t > 0 \quad (3.21)$$

It is well known that the conjugate prior to both the Poisson and exponential

Table 3.1: Properties of parameterized distributions relating to consensus

R. V.	Likelihood	Parameters	Conjugate Prior	Hyperparameters	Cons. Prior	Cons. Vars.
Continuous Value $\mathbf{x}$	normal	Unknown Mean $\mu$ , (Known Covariance $\Sigma$ )	normal	Mean $\hat{x}$ , and Covariance $P$	-	$Y = P^{-1} a$ $y = Y \mu$
	normal	Unknown Covariance $\Sigma$ , (Known Mean, $\mu$ )	inverse Wishart	Sample Covariance $\Psi$ and Counts $m$	-	$\Psi$ , $m$
Discrete Events $\mathbf{x}$	multinomial	Probability $\mathbf{p}$	Dirichlet	Counts $\alpha$	$\prod_{i=1}^m p_i^{-1}$	$\alpha$
	geometric	Probability $p$	beta	Counts $\alpha$	$p^{-1}$	$\alpha$
	binomial	Probability $p$	beta	Counts $\alpha$	$p^{-1}$	$\alpha$
	Bernoulli	Probability $p$	beta	Counts $\alpha$	$p^{-1}$	$\alpha$
Arrivals $k$	Poisson	Arrival Rate $\lambda$	gamma	Arrivals $\alpha$ , Time $\beta$	$\lambda^{-1}$	$\alpha$ , $\beta$
Inter-Arrival Time $t$	exponential	Arrival Rate $\lambda$	gamma	Arrivals $\alpha$ , Time $\beta$	$\lambda^{-1}$	$\alpha$ , $\beta$
Time for $\alpha$ Arrivals $t$	gamma	Rate $\beta$ (Known Arrivals $\alpha$ )	gamma	Arrivals $\alpha'$ , Time $\beta'$	$\beta^{-1}$	$\alpha'$ , $\beta'$
Size $x$	Pareto	Shape $k$ (Known Scale $x_m$ )	gamma	Arrivals $\alpha$ , Size $\beta$	$k^{-1}$	$\alpha$ , $\beta$

<sup>a</sup> $Y$  and  $y$  are the information variables as obtained using the information form of the Kalman filter and satisfy the additive update property

distributions is the gamma distribution

$$f_G(\lambda|\alpha, \beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda} \quad (3.22)$$

where  $\alpha$  is called the shape parameter, and  $\beta$  is the rate parameter. The mean and variance of the gamma distribution are found as:

$$\begin{aligned} \mu_\lambda &= \frac{\alpha}{\beta} \\ \sigma_\lambda^2 &= \frac{\alpha}{\beta^2} \end{aligned}$$

In terms of the Bayesian inference properties, the Poisson or exponential distribution would be the sampled Likelihood function, while the gamma distribution would be the form of the prior and posterior, with hyperparameters  $\alpha$  and  $\beta$ . Closed-form update equations of the hyperparameters are derived from the update step (shown here for sampling the Poisson distribution to obtain a measurement  $k$ ):

$$\begin{aligned} p(\lambda|\alpha, \beta, k) &\propto p(\lambda, k|\alpha, \beta, T) \\ &= f_P(k|\lambda, T) f_G(\lambda|\alpha, \beta) \\ &= \frac{(\lambda T)^k e^{-\lambda T}}{k!} \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\beta\lambda} \\ &\propto \lambda^{k+\alpha-1} e^{-(\beta+T)\lambda} \\ &\propto f_G(\lambda|\alpha + k, \beta + T) \end{aligned}$$

Thus, in the form of Eq. 3.8, the hyperparameter update is given by

$$\underbrace{\begin{bmatrix} \alpha \\ \beta \end{bmatrix}}_{\omega} \leftarrow \underbrace{\begin{bmatrix} \alpha \\ \beta \end{bmatrix}}_{\omega} + \underbrace{\begin{bmatrix} k \\ T \end{bmatrix}}_{h(z)}$$

These update equations can easily be transformed to account for sampling the inter-arrival time from the exponential likelihood by letting  $k = 1$  and  $T = t$ .

The selection of non-informative consensus prior can now be motivated through the application to the arrival rate parameter by extending Equation 3.10 to consider explicitly the probability distributions relevant to the arrival rate. In particular, an initial guess for a non-informative prior will be assumed to be gamma with parameters  $\alpha[0]_i$  and  $\beta[0]_i$ . Thus the centralized estimate becomes:

$$\begin{aligned}
p(\lambda|\alpha_{cent}, \beta_{cent}) &= \prod_i^N \left( \frac{f_G(\lambda|\alpha_i, \beta_i)}{f_G(\lambda|\alpha[0]_i, \beta[0]_i)} \right) f_G(\lambda|\alpha[0], \beta[0]) \\
&\propto \prod_i^N \left( \frac{\lambda^{\alpha_i-1} e^{-\beta_i \lambda}}{\lambda^{\alpha[0]_i-1} e^{-\beta[0]_i \lambda}} \right) \lambda^{\alpha[0]-1} e^{-\beta[0] \lambda} \\
&= \prod_i^N (\lambda^{\alpha_i-1-(\alpha[0]_i-1)} e^{-(\beta_i-\beta[0]_i)\lambda}) \lambda^{\alpha[0]-1} e^{-\beta[0] \lambda} \\
&= \left( \lambda^{\sum_i^N (\alpha_i-\alpha[0]_i)} e^{-\sum_i^N (\beta_i-\beta[0]_i)\lambda} \right) \lambda^{\alpha[0]-1} e^{-\beta[0] \lambda} \\
&= \lambda^{\sum_i^N (\alpha_i-\alpha[0]_i)+\alpha[0]-1} e^{-(\sum_i^N (\beta_i-\beta[0]_i)-\beta[0])\lambda} \\
&\propto f_G \left( \lambda \left| \sum_i^N (\alpha_i - \alpha[0]_i) + \alpha[0], \sum_i^N (\beta_i - \beta[0]_i) - \beta[0] \right. \right)
\end{aligned}$$

In this form it is apparent that the use of any non-zero prior values will remove information from the centralized estimate since all the information that each agent has is independent and contained entirely in the  $\alpha_i$ 's and  $\beta_i$ 's. This means that, for example, if an agent has seen 5 arrivals in the last 10 minutes but only communicates that it's seen 3 arrivals in the last 5 minutes, the other agents lose access to the other two data points and the system effectively loses information<sup>5</sup>.

Thus, in order for the centralized estimate to utilize all the information that is available, it may make sense to try using a prior distribution with  $\alpha[0] = \beta[0] = 0$  so as to avoid losing information. This selection unfortunately leads to an unrealizable prior since the corresponding gamma distribution will contain undefined values in the numerator ( $0^0$ ) and denominator ( $(-1)!$ ). While this prevents the explicit use of the

---

<sup>5</sup>The removal of *redundant* information is the primary motivation for channel filtering in the data fusion community, where redundant information is introduced by communication loops in the network and through correlation of measurements. One of the benefits of using consensus methods is that network-induced redundancy is eliminated through the form of the consensus protocol.

gamma distribution, we can instead consider an improper consensus prior<sup>6</sup> [47, 49] of the form  $\lambda^{-1}$  which will obtain the same end as setting the hyperparameters to zero. In particular, it is easy to show that, using this prior, the resulting consensus distribution will be the desired gamma distribution with the aggregation of all the individual agents' hyperparameters:

$$\begin{aligned}
p(\lambda|\alpha_{cent}, \beta_{cent}) &\propto \prod_i^N \left( \frac{\lambda^{\alpha_i-1} e^{-\beta_i \lambda}}{\lambda^{-1}} \right) \lambda^{-1} \\
&= \prod_i^N (\lambda^{\alpha_i} e^{-\beta_i \lambda}) \lambda^{-1} \\
&= \lambda^{\sum_i^N \alpha_i - 1} e^{-\sum_i^N \beta_i \lambda} \\
&\propto f_G \left( \lambda \left| \sum_i^N \alpha_i, \sum_i^N \beta_i \right. \right) \tag{3.23}
\end{aligned}$$

Finally, the centralized parameter estimate for the arrival rate, to be denoted  $\lambda_B$ , is given in Eq. 3.24 and is achieved by taking the ratio of the sums of the individual hyperparameters.

$$\lambda_B = \frac{\sum_{i=1}^N \alpha_i}{\sum_{i=1}^N \beta_i} \tag{3.24}$$

This last equation can also be motivated intuitively by considering two agents that have independent hyperparameters  $\alpha$  equal to 4 and 6 and  $\beta$  equal to 2 and 8, respectively. These hyperparameters are roughly equivalent to saying that the agents have seen 4 and 6 arrivals in 2 and 8 minutes, respectively, such that the local estimates of lambda are  $4/2 = 2$  and  $6/8 = 0.75$ . Combined, however, the two agents have observed a total of 10 arrivals in 10 minutes, such that the centralized estimate should be  $10/10 = 1$ , which is the value that would be obtained using Eq. 3.24.

---

<sup>6</sup>Improper priors are priors that are not probability distributions in their own right (and therefore provide no meaningful a priori information on the parameter) but serve to produce a meaningful or desired posterior distribution. They are often used as non-informative priors because of their flexibility and ability to produce unbiased posterior distributions of a desired form.



## Results and Discussion

The following simulation results utilize the same initial conditions as those used in the motivating examples of average and Kalman consensus trajectories in Figures 1-1 and 1-2. In particular, five agents are trying to reach consensus on the arrival rate parameter, where the initial conditions are given in Table 3.2.

Table 3.2: Initial conditions for  $\lambda$  consensus

Agent	$\alpha_i$	$\beta_i$	$\mu_{\lambda i}$	$\sigma_{\lambda i}^2$
1	5	5	1	0.2
2	9	3	3	1
3	12	2	6	3
4	14	2	7	3.5
5	16	2	8	4

The resulting centralized estimate for these initial conditions is  $\alpha = 56$ ,  $\beta = 14$ , with a resulting MMSE estimate of  $\lambda = 4$  and variance of  $\sigma_\lambda^2 \approx 0.286$ . To finalize the set-up, the parameter and hyperparameter consensus algorithms both use edge weights,  $a_{ij}$ , set to  $\frac{1}{N}$  for  $i \neq j$ ,  $(i, j) \in \mathcal{E}$ , and  $a_{ii} = 1 - \sum_{j \neq i}^N a_{ij}$ . These weights are not optimized for any convergence speed metric nor do they have any particular impact on the convergence value, but are merely weights that allow for the desired result to be shown. Further, two five-agent networks will be used to highlight convergence on balanced and unbalanced graphs. Figure 3-1(a) shows a directed, balanced double-ring network, while Figure 3-1(b) shows a biased network where agent 5 is able to talk to everyone. The latter network would result in an uncompensated steady-state consensus estimate of:

$$\lim_{k \rightarrow \infty} \xi_i[k] = \begin{bmatrix} \frac{1}{16} & \frac{1}{16} & \frac{1}{8} & \frac{1}{4} & \frac{1}{2} \end{bmatrix} \xi[0]$$

### Consensus on Initial Conditions

Figure 3-2 shows the resulting parameter trajectory while agents are coming to consensus using the hyperparameter consensus method on the double-ring graph in Fig-

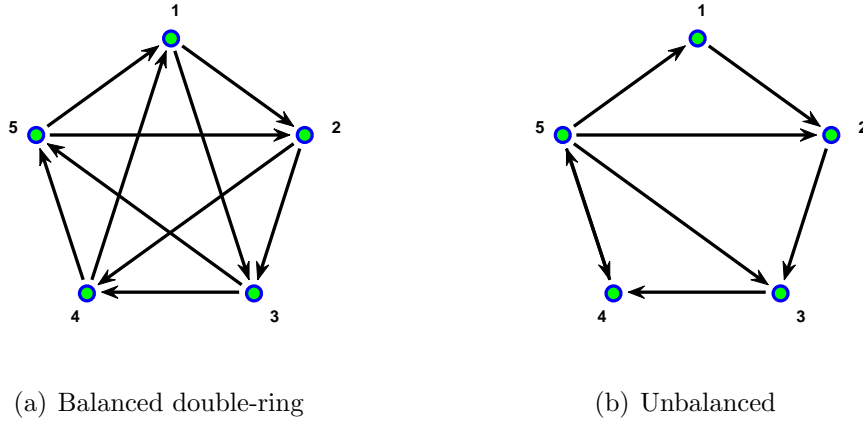
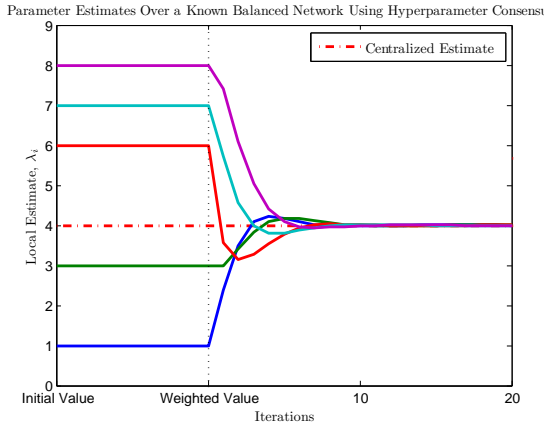


Figure 3-1: Five agent connectivity graphs

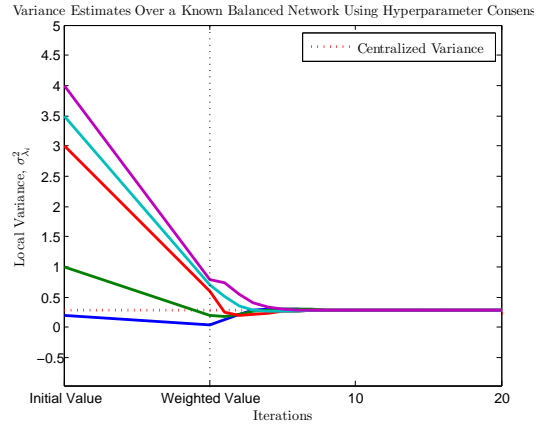
ure 3-1(a). Recall that, for sum consensus, it is required to weight the initial conditions to achieve the desired result. This weighting is shown in the transition from “Initial Value” to “Weighted Value” (equivalent to the step in Eq. 3.12 from  $k = 0^-$  to  $k = 0$ ), after which the consensus proceeds normally. As is expected, the MMSE estimate and associated variance converge exactly to the centralized values. The sum-consensuses trajectories for  $\alpha$  and  $\beta$  are shown in Figures 3-3(a) and 3-3(b).

Inspecting the transient period before convergence, it is apparent that the weighting of the hyperparameters quite significantly affects the variance of the agents, though the MMSE parameter estimate remains unchanged. Since, in most cases, the parameter estimate is of primary concern and is unchanged by the weighting, this discontinuity can generally be ignored.

The previous results were shown for a balanced double-ring network, where each agent talks to the next two agents in the ring. The same results can be obtained when the network is known and unbalanced by selecting alternate weights. Figure 3-4 shows the same consensus but now on the network in Figure 3-1(b), which is biased towards agent 5 (purple). Similarly, the hyperparameter trajectories are shown in Figure 3-5. As required, the purple values denoting agent 5’s hyperparameters are scaled much lower than the other agents, which allows the system to converge to the centralized parameter estimate.

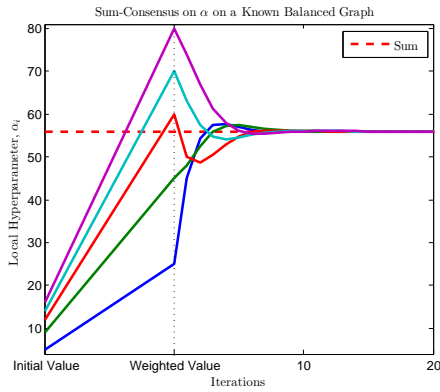


(a) Mean Value  $\mathbf{p}$

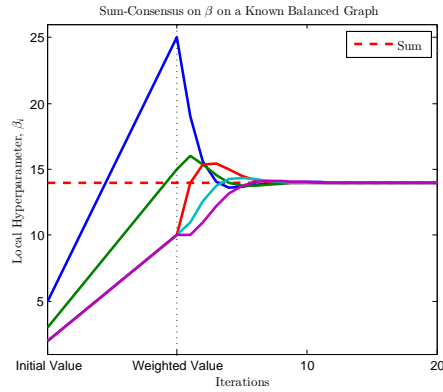


(b) Variance  $\sigma_{\mathbf{p}}^2$

Figure 3-2: Local parameter estimate and variance achieved by hyperparameter consensus on a balanced network



(a)  $\alpha$



(b)  $\beta$

Figure 3-3: Hyperparameter trajectories during consensus on a balanced network

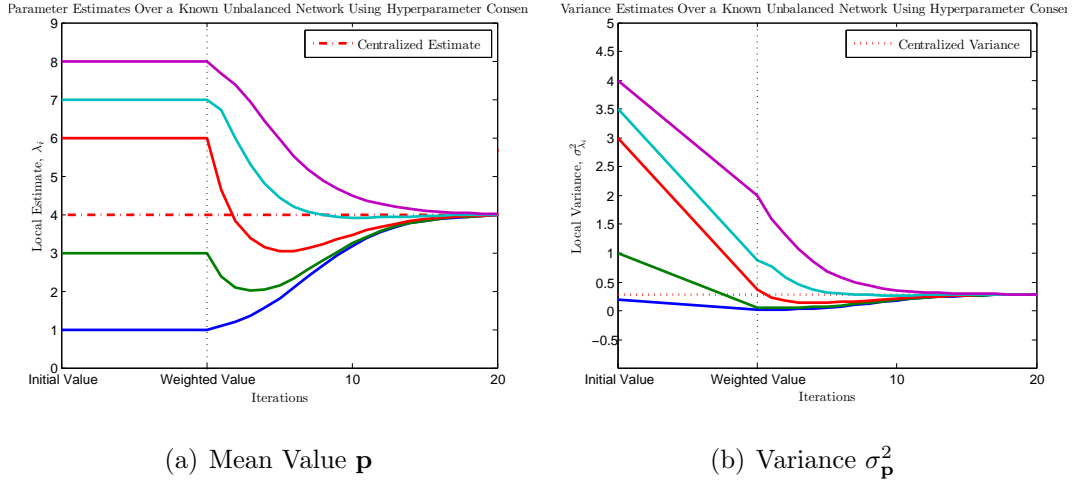


Figure 3-4: Local estimate and variance achieved by hyperparameter consensus on  $\lambda$  over a known, unbalanced network

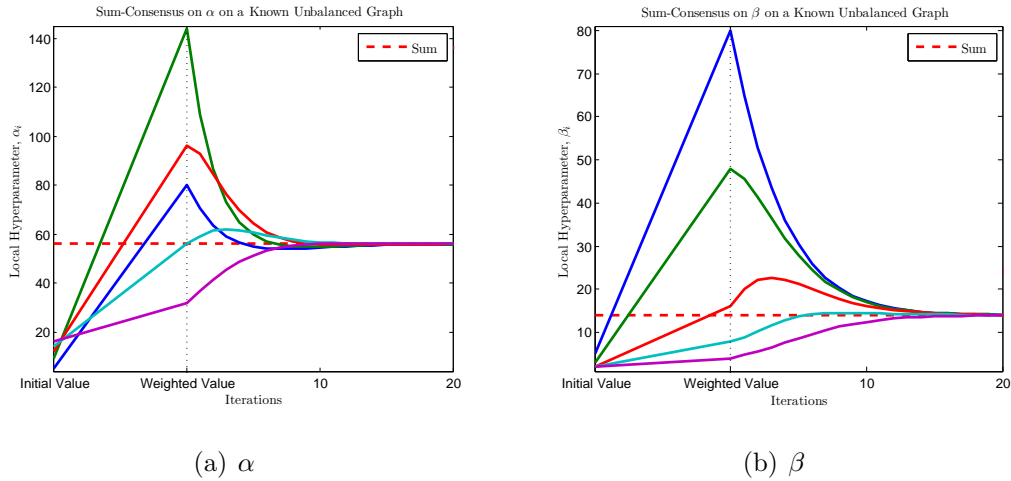


Figure 3-5: Hyperparameter trajectories during consensus on  $\lambda$  over a known, unbalanced network

## With Measurements

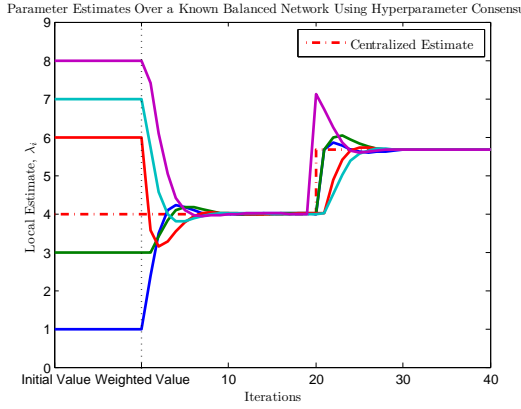
This section graphically displays the result of concurrent measurement during the consensus phase as outlined in Theorem 3.1.2. Figure 3-6 shows this result on a balanced network where the measurement is taken after consensus has effectively been reached, while Figure 3-7 shows the impact of taking a measurement during the transient period. Both cases show the desired convergence to the augmented centralized estimate from a measurement made by agent 5 of  $\alpha_{meas} = 80$  and  $\beta_{meas} = 10$ . The hyperparameter trajectories for the second case are shown in Figure 3-8.

### 3.2.2 Unknown Network

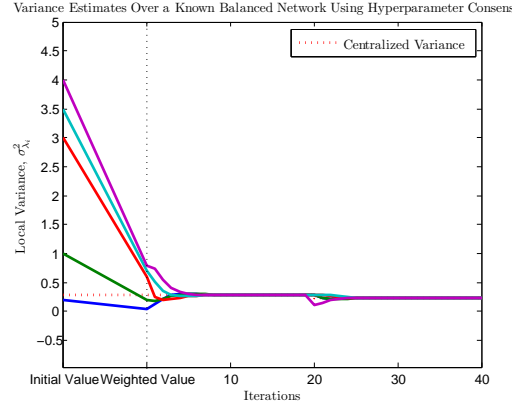
**Conjecture:** *On arbitrary unknown networks, agents running consensus on the hyperparameters are expected to achieve a steady-state parameter estimate that is closer to the centralized Bayesian estimate as compared to agents running the same consensus protocol on the parameter values themselves. Further, it is also expected that the variance in this incurred error will be less in the hyperparameter consensus case than in the parameter consensus case.*

To justify this conjecture, an example problem will be constructed to determine the expected results: Consider a group of  $N$  agents, each of which has an initial local estimate of  $\alpha$ ,  $\beta$ , and  $\lambda$  obtained through sampling distributions on  $\alpha$  and  $\beta$ . The  $\alpha$  distribution is assumed to be uniform over a discrete set  $[a, b]$ , while  $\beta$  is given as the sum of  $\alpha$  random variables  $X_i$ , each of which are distributed exponentially with parameter  $\lambda$ . The sum of  $\alpha$  independent and identically distributed exponential random variables is a random variable defined by the Erlang distribution:

$$f_E(\beta|\alpha, \lambda) = \frac{\lambda^\alpha \beta^{\alpha-1} e^{-\lambda\beta}}{(\alpha - 1)!}$$

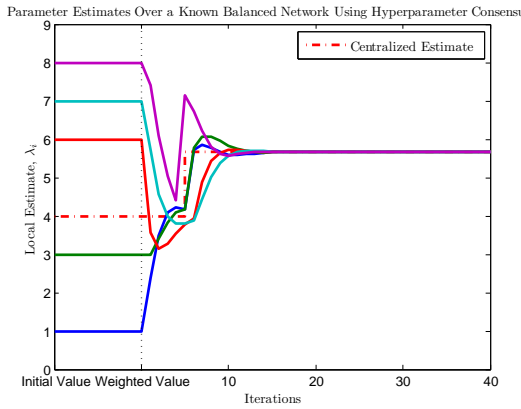


(a) Mean Value  $\mathbf{p}$

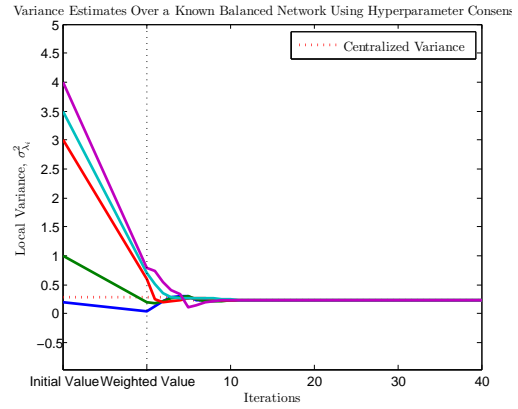


(b) Variance  $\sigma_{\mathbf{p}}^2$

Figure 3-6: Local estimate and variance achieved by hyperparameter consensus for  $\lambda$  with a measurement at  $k=20$



(a) Mean Value  $\mathbf{p}$



(b) Variance  $\sigma_{\mathbf{p}}^2$

Figure 3-7: Local estimate and variance achieved by hyperparameter consensus for  $\lambda$  with a measurement at  $k=5$

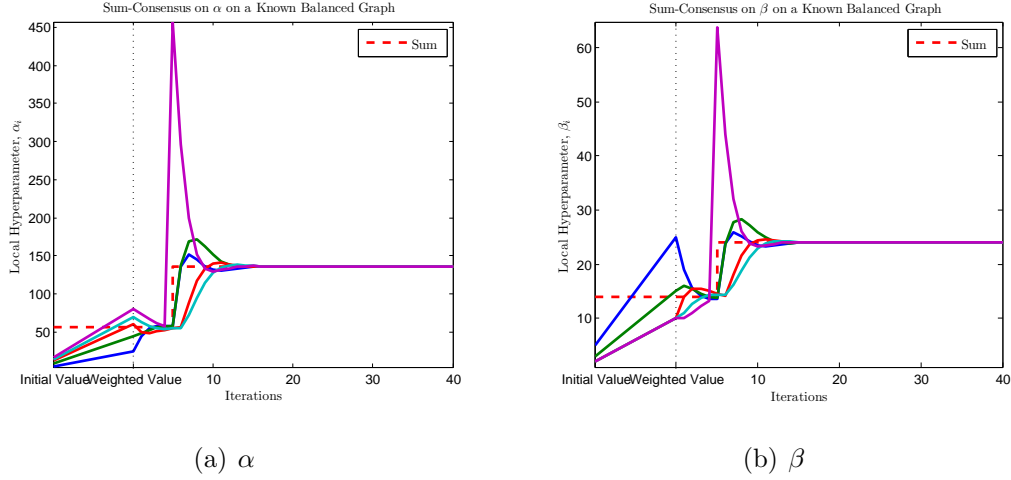


Figure 3-8: Hyperparameter trajectories during consensus on  $\lambda$  over a balanced network with a measurement at  $k=5$

Thus, the joint distribution for a local  $\alpha_i$  and  $\beta_i$  is given by:

$$\begin{aligned}
 p(\alpha_i, \beta_i | \lambda) &= p(\beta_i | \alpha_i, \lambda) p(\alpha_i) \\
 &= \frac{\lambda^\alpha \beta^{\alpha-1} e^{-\lambda\beta}}{(\alpha-1)!} \frac{1}{b-a+1}
 \end{aligned}$$

All the agents are assumed to have independent beliefs, so the joint distribution over  $\alpha = [\alpha_1, \dots, \alpha_N]$  and  $\beta = [\beta_1, \dots, \beta_N]$  becomes:

$$p(\alpha, \beta | \lambda) = \prod_{i=1}^N p(\beta_i | \alpha_i, \lambda) p(\alpha_i)$$

Each agent maintains a local estimate  $\lambda_i = \alpha_i / \beta_i$ , and the group of agents come to consensus over a fixed network which will provide a consensus value of  $\lim_{k \rightarrow \infty} \xi_i(k) = \nu^T \xi(0)$ .

It is assumed that the agents are unaware of the network topology, and, therefore, it is not possible to evaluate Eq. 3.12, so no scaling of the initial results occurs.

Further, it is also required that the selected consensus protocol is independent of any knowledge of the network. Thus, the following results will be defined using the

consensus update

$$\xi_i[k+1] = \xi_i[k] + \epsilon \sum_{j \in \mathcal{I}_i} (\xi_j[k] - \xi_i[k]) \quad \forall i \quad (3.25)$$

where  $\epsilon = 1/N$  and  $\mathcal{I}_i$  denotes the set of all incoming neighbors,  $j$ , such that  $(j, i) \in \mathcal{E}$ . This is an equivalent form to Equations 2.2 and 3.11, though does not require explicit knowledge of the network structure.

It is desired to compare the error obtained when using the consensus protocol in Eq. 3.25 on the parameter  $\lambda_i$  alone versus the steady-state error obtained by running the protocol on the hyperparameters  $\alpha_i$  and  $\beta_i$ . To achieve this, define the following error variables, where  $\bar{\lambda}$  denotes the steady-state estimate using parameter consensus,  $\hat{\lambda}$  denotes the steady-state estimate using hyperparameter consensus, and  $\lambda_B$  denotes the centralized Bayesian estimate:

$$\begin{aligned} \text{Parameter Error: } \bar{e} &= \left| \frac{\bar{\lambda} - \lambda_B}{\lambda_B} \right| = \left| \frac{\bar{\lambda}}{\lambda_B} - 1 \right| \\ &= \left| \frac{\sum_i \nu_i \frac{\alpha_i}{\beta_i}}{\frac{\sum_i \alpha_i}{\sum_i \beta_i}} - 1 \right| \\ \text{Hyperparameter Error: } \hat{e} &= \left| \frac{\hat{\lambda} - \lambda_B}{\lambda_B} \right| = \left| \frac{\hat{\lambda}}{\lambda_B} - 1 \right| = \left| \frac{\frac{\sum_i \nu_i \alpha_i}{\sum_i \nu_i \beta_i}}{\frac{\sum_i \alpha_i}{\sum_i \beta_i}} - 1 \right| \\ &= \left| \left( \frac{\sum_i \nu_i \alpha_i}{\sum_i \alpha_i} \right) \left( \frac{\sum_i \beta_i}{\sum_i \nu_i \beta_i} \right) - 1 \right| \end{aligned}$$



Thus, we can find an expression for the expected value of these errors:

$$\begin{aligned}
E[\bar{e}(\alpha, \beta)] &= \sum_{\alpha} \int_{\beta} \bar{e}(\alpha, \beta) p(\alpha, \beta | \lambda) d\beta \\
&= \sum_{\alpha} \int_{\beta} \left| \frac{\sum_i \nu_i \frac{\alpha_i}{\beta_i}}{\frac{\sum_i \alpha_i}{\sum_i \beta_i}} - 1 \right| \prod_{i=1}^N \frac{\lambda^{\alpha} \beta^{\alpha-1} e^{-\lambda \beta}}{(\alpha-1)!} \frac{1}{b-a+1} d\beta \quad (3.26)
\end{aligned}$$

$$\begin{aligned}
E[\hat{e}(\alpha, \beta)] &= \sum_{\alpha} \int_{\beta} \hat{e}(\alpha, \beta) p(\alpha, \beta | \lambda) d\beta \\
&= \sum_{\alpha} \int_{\beta} \left| \left( \frac{\sum_i \nu_i \alpha_i}{\sum_i \alpha_i} \right) \left( \frac{\sum_i \beta_i}{\sum_i \nu_i \beta_i} \right) - 1 \right| \prod_{i=1}^N \frac{\lambda^{\alpha} \beta^{\alpha-1} e^{-\lambda \beta}}{(\alpha-1)!} \frac{1}{b-a+1} d\beta \quad (3.27)
\end{aligned}$$

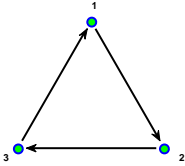
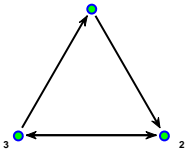
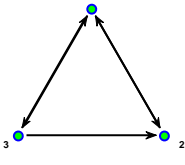
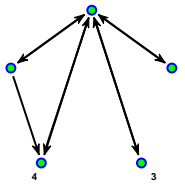
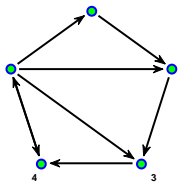
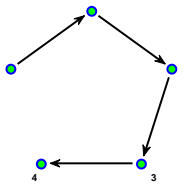
Equations 3.26 and 3.27 have no closed form solution and need to be integrated numerically. The approach taken here is to convert the integral to a discrete sum using Monte-Carlo sampling. The distribution for  $\alpha$  and  $\beta$  is sampled  $N_s = 200,000$  times, each sample giving a value of  $\hat{e}$  and  $\bar{e}$ . The integral is then approximated by summing the resulting sets for each of the errors. Table 3.3 shows the results for a several different 3- and 5-agent networks<sup>7</sup>. The apparent trend is that, in all cases, the expected error using purely parameter consensus is quite far off of the desired Bayesian centralized estimate and much higher than the error incurred using hyperparameter consensus. Further, the standard deviation in the parameter error is much larger than that for the hyperparameter error. This suggests that not only does hyperparameter consensus perform better on average, but is expected to do better in worst-case scenarios too.

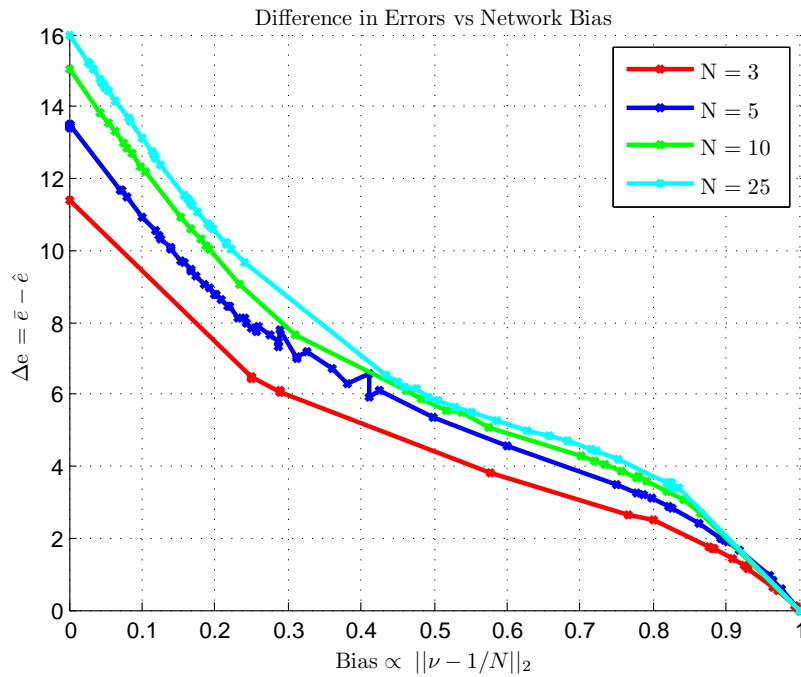
Figures 3-9(a) and 3-9(b) show the expected performance gap between the parameter average and hyperparameter consensus methods. The x-axis of both is a measure of the bias of the network, which is determined by taking the scaled 2-norm of the true consensus eigenvalue,  $\nu$ , minus the desired average consensus eigenvalue,  $\mathbf{e}/N$ . The scaling factor is such that  $\nu = [0, \dots, 0, 1]^T$  gives a bias score of 1. Thus, a bias of 0 means that the the network is unbiased and all agents are given equal weight, while a bias near 1 means that one agent's information will naturally dominate the

---

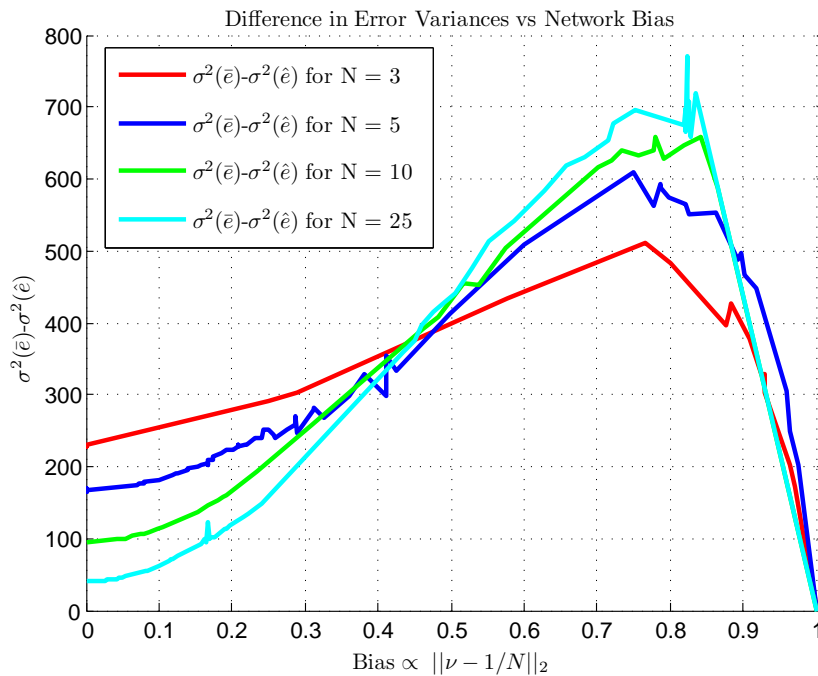
<sup>7</sup>All results shown for the MC simulations were aggregated over 20 trials, resulting in a sample variance for each metric of less than 1% of the mean value.

Table 3.3: Expected consensus error comparison with initial conditions

Network Structure	$\nu$	Bias	$\alpha \in [a, b]$	$E[\bar{e}]$ (%)	$\sigma_{\bar{e}}$ (%)	$E[\hat{e}]$ (%)	$\sigma_{\hat{e}}$ (%)
	$\begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}$	0	[5 10]	11.40	15.12	0	0
	$\begin{bmatrix} 1/4 \\ 1/4 \\ 1/2 \end{bmatrix}$	0.25	[5 10]	12.35	17.63	5.87	4.33
	$\begin{bmatrix} 1/3 \\ 1/6 \\ 1/2 \end{bmatrix}$	0.289	[5 10]	12.93	18.20	6.85	5.41
	$\begin{bmatrix} 1/5 \\ 1/5 \\ 1/5 \\ 1/10 \\ 3/10 \end{bmatrix}$	0.158	[5 10]	13.71	14.60	4.06	3.19
	$\begin{bmatrix} 1/16 \\ 1/16 \\ 1/8 \\ 1/4 \\ 1/2 \end{bmatrix}$	0.412	[5 10]	16.74	20.66	10.84	8.56
	$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$	1	[5 10]	31.81	37.87	31.81	37.87



(a) Expected difference in errors ( $E(\bar{e}) - E(\hat{e})$ )



(b) Difference in variance of errors ( $\sigma^2(\bar{e}) - \sigma^2(\hat{e})$ )

Figure 3-9: Monte-Carlo error results obtained over an unknown network

network (a bias equal to 1 implies that the network is no longer strongly connected but, instead, that one agent receives no incoming messages and so the other agents must converge to its estimate). The y-axis of Figure 3-9(a) shows the expected error difference

$$E[\Delta e] = E[\bar{e}] - E[\hat{e}]$$

such that positive values imply a larger expected error using parameter consensus than with hyperparameter consensus. Figure 3-9(b) plots the difference in error variances:

$$\sigma_{\bar{e}}^2 - \sigma_{\hat{e}}^2$$

In both graphs, positive values show that the respective hyperparameter metric (error or variance) is *smaller* than the parameter consensus value.

The results suggest that attempting an average hyperparameter consensus method using Eq. 3.25 is expected to incur less error than running the same consensus on the parameter  $\lambda$  itself, especially if the network is nearly balanced. In the cases where the network is badly biased towards a particular agent's information (bias close to 1), the two errors approach the same value as that agent begins to dominate both the hyperparameter and parameter steady-state results. A similar trend also holds in terms of the variances of the errors, as shown in Figure 3-9(b). The variance in the hyperparameter consensus error was found to be smaller than the variance in the parameter error for all cases except when the network has a bias metric of 1, where the results are equivalent. This suggests that not only is the resulting hyperparameter consensus value more *accurate*, it is also more *precise* than the parameter consensus method.

### 3.2.3 The Dirichlet Prior

The Dirichlet distribution is commonly used in the Bayesian estimation community to represent the current estimate of a probability vector [50–52] since it maintains a distribution over a vector lying in the unit simplex. In a similar framework, the

distribution is presented here as a means for agents to maintain and communicate an uncertain estimate of a probability vector,  $\mathbf{p} \in \mathfrak{R}^m$ , across a network. Though the use of the Dirichlet can be extended to other vectors with similar properties, such as a convex vector of weights, primary focus will be given here to the application of the distribution to probability vectors.

The Dirichlet distribution [47], denoted here specifically as  $f_D(\cdot)$ , is the multi-variate extension of the beta distribution, and can therefore be extended to the beta distribution prior, as well as the Bernoulli, binomial, and geometric likelihoods with little difficulty. It is parameterized by a vector of *counts*,  $\alpha_i \geq 1$ , and has the form:

$$\begin{aligned} f_D(\mathbf{p}|\alpha) &= K \prod_{i=1}^m p_i^{\alpha_i-1}, \quad \sum_i p_i = 1 \\ &= K p_1^{\alpha_1-1} p_2^{\alpha_2-1} \dots \left(1 - \sum_{i=1}^{m-1} p_i\right)^{\alpha_m-1} \end{aligned} \quad (3.28)$$

where  $K$  is a normalizing factor.

The MMSE estimate of  $\mathbf{p}$  is given by the mean value of the Dirichlet,  $\bar{\mathbf{p}}$ . The mean and variance of each element of the probability vector can be calculated as follows, where  $\alpha_0 = \sum_{i=1}^m \alpha_i$ :

$$\bar{p}_i = \frac{\alpha_i}{\alpha_0} \quad (3.29)$$

$$\Sigma_{ij} = \begin{cases} \frac{\alpha_i(\alpha_0 - \alpha_i)}{\alpha_0^2(\alpha_0 + 1)} & \text{if } i = j \\ \frac{-\alpha_i \alpha_j}{\alpha_0^2(\alpha_0 + 1)} & \text{otherwise} \end{cases} \quad (3.30)$$

The Dirichlet distribution is conjugate to the multinomial likelihood [47],  $f_M(\cdot)$ , which depicts the probability of observing  $x_i$  occurrences of event  $i$ :

$$f_M(\mathbf{x}|\mathbf{p}) = K \prod_{i=1}^m p_i^{x_i}, \quad x_i \geq 0 \quad (3.31)$$

The Bayesian update gives a closed form solution to update the hyperparameters as:

$$\begin{aligned}
f_D(\mathbf{p}|\alpha, x) &\propto f_D(\mathbf{p}|\alpha) f_M(\mathbf{x}|\mathbf{p}) \\
&= \prod_{i=1}^m p_i^{\alpha_i-1} p_i^{x_i} = \prod_{i=1}^m p_i^{\alpha_i+x_i-1} \\
&\Rightarrow \alpha_i \leftarrow \alpha_i + x_i \quad \forall i \in (1, \dots, m)
\end{aligned} \tag{3.32}$$

It may be helpful to some to motivate the additive form of the centralized hyperparameter estimate by considering an intuitive example: Two agents are actively estimating the bias of an unfair coin. Agent 1 has observed 3 heads out of 4 tries, and, with no initial information on the probabilities (i.e. uses an uninformative prior), its best estimate is the MLE,  $p_{heads} = 3/4 = 0.75$ . Agent 2 has, independently, observed 5 heads out of 20 tries, and believes the probability is  $p_{heads} = 5/20 = 0.25$ . Between the two, they have observed 8 heads out of 24 tries, which should result in an estimate (under the same assumption of no initial knowledge) of  $p_{heads} = 8/24 \approx 0.333$ .

Again, appealing to the similarity to the conjugate distribution with null hyperparameters,  $f_D(\mathbf{p}|\mathbf{0})$ , the consensus prior is defined as

$$p(\mathbf{p}) = \prod_{i=1}^m p_i^{-1}$$

The derivation of the centralized estimate for the above example is shown below:

$$\begin{aligned}
f_D(\mathbf{p}|\alpha) &\propto \prod_{i=1}^2 \left( \frac{p_i(\mathbf{p}|\alpha_i)}{p_i(\mathbf{p})} \right) p(\mathbf{p}) \\
&= \frac{p_1(\mathbf{p}|\alpha_1) p_2(\mathbf{p}|\alpha_2)}{p_1(\mathbf{p}) p_2(\mathbf{p})} p(\mathbf{p}) \\
&\propto \frac{p_1^{\alpha_1^{heads}-1} p_2^{\alpha_1^{tails}-1} p_1^{\alpha_2^{heads}-1} p_2^{\alpha_2^{tails}-1}}{p_1^{-1} p_2^{-1} p_1^{-1} p_2^{-1}} p_1^{-1} p_2^{-1} \\
&= p_1^{\alpha_1^{heads} + \alpha_2^{heads}} p_2^{\alpha_1^{tails} + \alpha_2^{tails}} p_1^{-1} p_2^{-1} \\
&= p_1^{\alpha_1^{heads} + \alpha_2^{heads} - 1} p_2^{\alpha_1^{tails} + \alpha_2^{tails} - 1} \\
&= f_D(\mathbf{p}|\alpha_1 + \alpha_2)
\end{aligned}$$

When using the Dirichlet, it is important to ensure that the agents have at least one observation of each outcome in order for the resulting posterior distribution to be a valid probability distribution. If this is not the case, it may be necessary to either assume a different prior distribution or implement a hierarchical Dirichlet process where only observed states are considered in the probability vector [53].

## Comparison to Kalman Consensus

Unlike the gamma prior, the Dirichlet prior and the problem of consensus on probabilities has many close parallels to existing methods. Primarily, under some assumptions, the Dirichlet distribution can closely resemble the normal distribution, suggesting that the Kalman consensus algorithms may provide a reasonable consensus approximation.

This similarity between the Dirichlet and normal distributions occurs in the limit as  $\alpha_0 \rightarrow \infty$  and only if  $\alpha_1 \approx \alpha_2 \approx \dots \approx \alpha_m$ . In other words, as the total number of observations grows large and if all outcomes are approximately equally probable, then the Dirichlet distribution can be roughly approximated by a multivariate normal with mean and variance as defined in Eq. 3.29 and 3.30. Further, [7] derived a closed-form update recursion for the mean and variance of the Dirichlet after observing a single measurement of outcome  $i'$ :

$$\bar{p}_i[k+1] = \bar{p}_i[k] + \sum_{ii'} \frac{\delta_{ii'} - \bar{p}_i[k]}{\bar{p}_i[k](1 - \bar{p}_i[k])} \quad (3.33)$$

$$\Sigma_{ij}^{-1}[k+1] = \begin{cases} \frac{\bar{p}_i[k](1 - \bar{p}_i[k])\Sigma_{ii}^{-1}[k] + 1}{\bar{p}_i[k+1](1 - \bar{p}_i[k+1])} & \text{if } i = j \\ \frac{\bar{p}_i[k]\bar{p}_j[k]\Sigma_{ij}^{-1}[k] - 1}{\bar{p}_i[k+1]\bar{p}_j[k+1]} & \text{otherwise} \end{cases} \quad (3.34)$$

The closed form mean and variance updates in equations 3.33 and 3.34, combined with the normal approximation, motivate the possible use of the Kalman consensus method for agreement on the Dirichlet distribution in [54]<sup>8</sup>. While this may be a reasonable approximation under the stated assumptions, the assumptions themselves

---

<sup>8</sup>Ref. [54] notes that special consideration is required when using the KC algorithm with the moments of the Dirichlet since the covariance matrix has zero row- and column-sums and is therefore not invertible. To compensate, the KC algorithm is only applied to a  $\{m-1 \times m-1\}$  submatrix of  $\Sigma$  and the  $\{m-1 \times 1\}$  subvector of  $\bar{p}$

Table 3.4: Initial conditions for  $\alpha$  consensus

Agent	$\alpha_1$	$\alpha_2$	$\alpha_3$	$p_1$	$p_2$	$p_3$
1	13	3	1	0.76	0.18	0.06
2	1	3	3	0.14	0.43	0.43
3	10	1	8	0.53	0.05	0.42
4	11	2	4	0.65	0.12	0.23
5	5	3	5	0.38	0.24	0.38
Centralized	40	12	21	0.55	0.16	0.29

(large number of counts and approximately equally likely outcomes) are fairly restrictive and are not appropriate in all situations. However, due to this possible similarity, Kalman methods will also be applied in the Dirichlet framework to evaluate just how biased an approximation they provide.

## Results and Discussion

The following sections provide simulation results for the hyperparameter and Kalman consensus methods achieving convergence on uncertain probability distributions. For all the cases shown, the initial conditions are outlined in Table 3.4, where  $\alpha_i$  represents the number of counts for outcome  $i$ . The centralized values are shown in the final row.

The consensus algorithms will be used as introduced in Sections 2.3.3 and 3.1, with the same communication graphs and graph weights as used for the gamma examples.

### Consensus on Initial Conditions

The parameter trajectories from the consensus problem are shown in Figures 3-10(a) and 3-10(b) for the hyperparameter and Kalman consensus methods, respectively, on a balanced network. Again, the hyperparameter consensus method converges to the exact centralized parameter result, with the  $\alpha$  trajectories shown in Figure 3-12. The Kalman consensus approach, however, is obviously biased from the centralized value. Further, the error in the local covariance matrices can be represented by the matrix 2-norm of the difference, where the 2-norm, or spectral norm, is defined for a square



matrix  $A$  as the square root of largest singular value of  $A$ :

$$\|A\|_2 = \sqrt{\bar{\sigma}(A)}$$

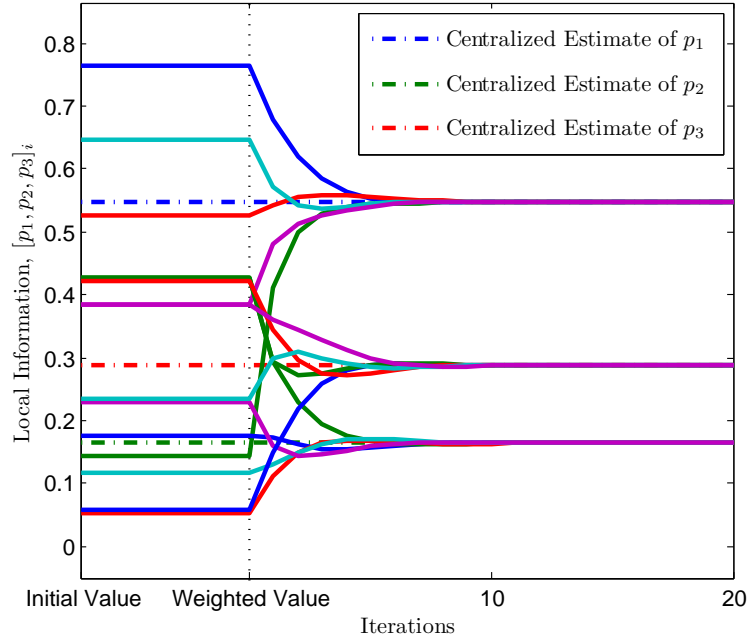
Figure 3-11 shows the ratio of the spectral norm of the difference between the local and centralized covariance matrices to the spectral norm of the centralized variance:

$$\frac{\|\Sigma_{local} - \Sigma_{cent}\|_2}{\|\Sigma_{cent}\|_2}$$

The error for the hyperparameter consensus, shown in dashed lines, steadily decreases as the hyperparameter-based estimate converges to the true centralized value. The error for the local covariance matrices obtained through Kalman consensus, however, converges to a constant ratio of 0.391 suggesting a steady-state bias in the variance as well. Therefore, despite theoretical similarities between the normal and Dirichlet distributions, the resulting steady-state Kalman consensus parameters suggest that the Kalman consensus filter is not accurate in this situation. When the minimum number of counts for each agent gets large and with roughly equivalent counts for each outcome (therefore satisfying the assumptions for the normal approximation) the problem itself becomes rather trivial since each agent has an estimate of  $\mathbf{p} \approx \mathbf{e}/m$ . Thus, any reasonable consensus algorithm will perform well due to the fact that the differences between the estimates is so small (though the hyperparameter method is still the only one to guarantee the desired convergence). It is then concluded that the use of the Kalman filter for approximate consensus on the Dirichlet probabilities is not robust to varied initial conditions and is therefore unadvised.

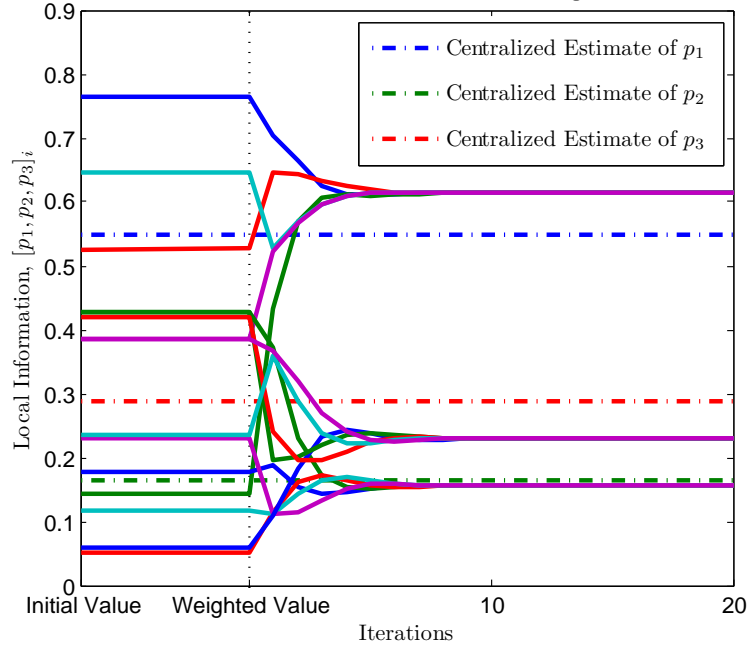
In the case of hyperparameter consensus over the known, unbalanced network shown in Figure 3-1(b), the agents are still able to achieve agreement, as evidenced in Figures 3-13 through 3-19. If attempting average consensus on the probabilities directly, a cautionary result is highlighted in Figure 3-15, where, due to the scaling required to achieve average consensus, the transient behavior of the local probabilities takes them outside of the unit simplex. Therefore, using parameter consensus on probabilities themselves is not recommended since the variables are not guaranteed

Local Estimates Over a Known Balanced Network Using Hyperparameter Consen



(a) Hyperparameter consensus method

Local Estimates Over a Known Balanced Network Using Kalman Consensus



(b) Kalman consensus method

Figure 3-10: Local parameter estimate achieved for  $\mathbf{p}$  on a balanced network

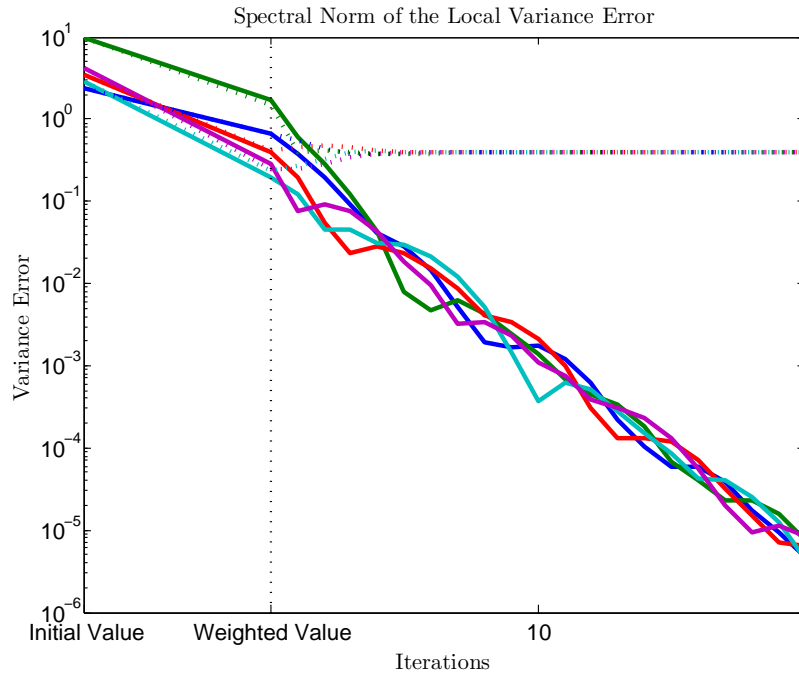


Figure 3-11: Local covariance error measured by  $\frac{\|\Sigma_{local} - \Sigma_{cent}\|_2}{\|\Sigma_{cent}\|_2}$  for the hyperparameter consensus (solid) and Kalman consensus (dotted) on a balanced network

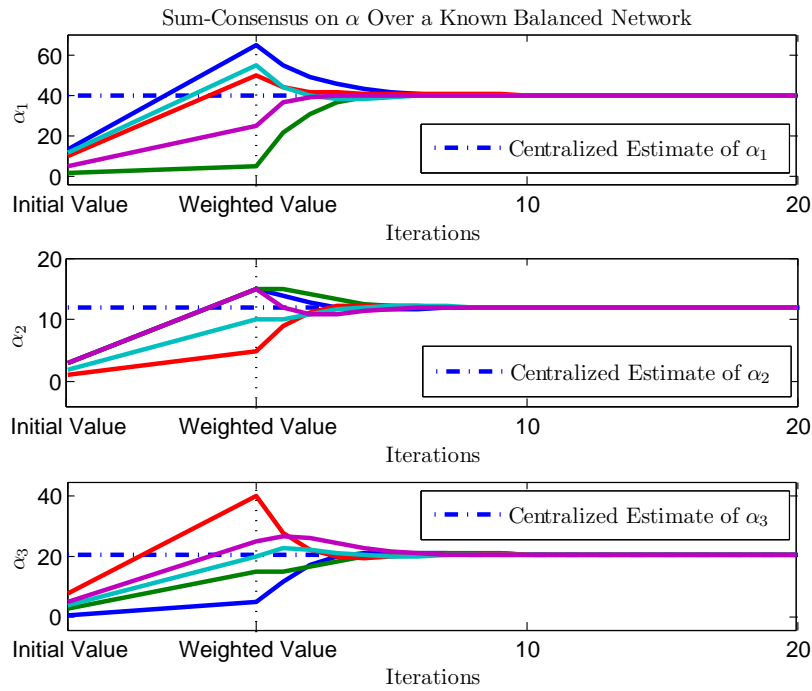


Figure 3-12: Hyperparameter trajectories during consensus on  $\mathbf{p}$  over a balanced network

to remain within the simplex, as well as the fact that they are not likely to converge to the desired result in the first place.

### **With Measurements**

This section highlights the effects of measurements on the consensus variables. The measurements are obtained by sampling from a multinomial distribution, such as observing the outcome of a probabilistic process (ie. toss of a die), possibly several times before incorporation into the estimate. Given a measurement by agent 4 (teal) of  $x^4 = [0 \ 1 \ 3]^T$  and 5 (purple) of  $x^5 = [1 \ 0 \ 2]^T$ , the hyperparameter consensus is, again, able to maintain convergence properties to the new centralized result.

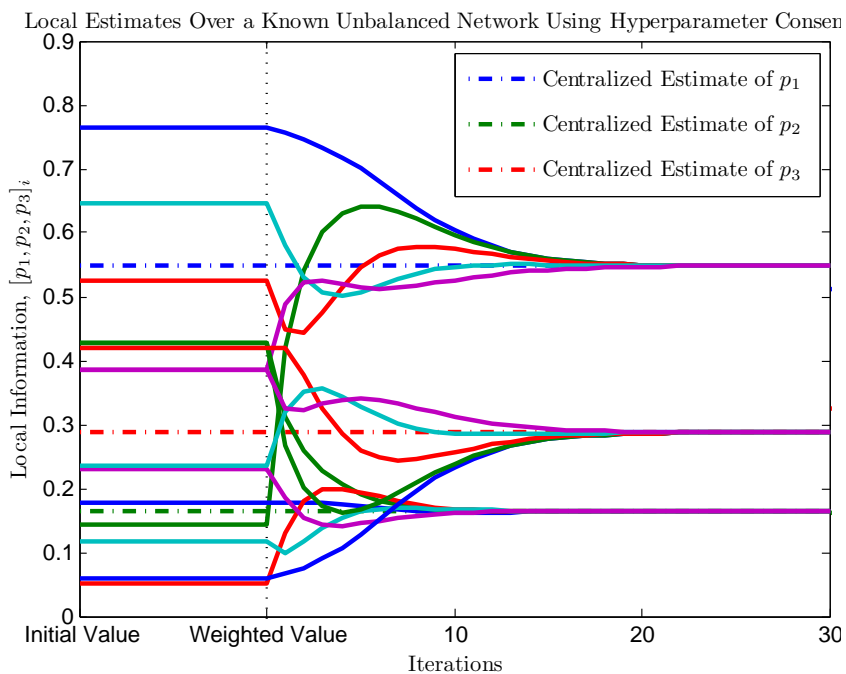


Figure 3-13: Local estimate achieved by the hyperparameter consensus method on  $\mathbf{p}$  over a known, unbalanced network

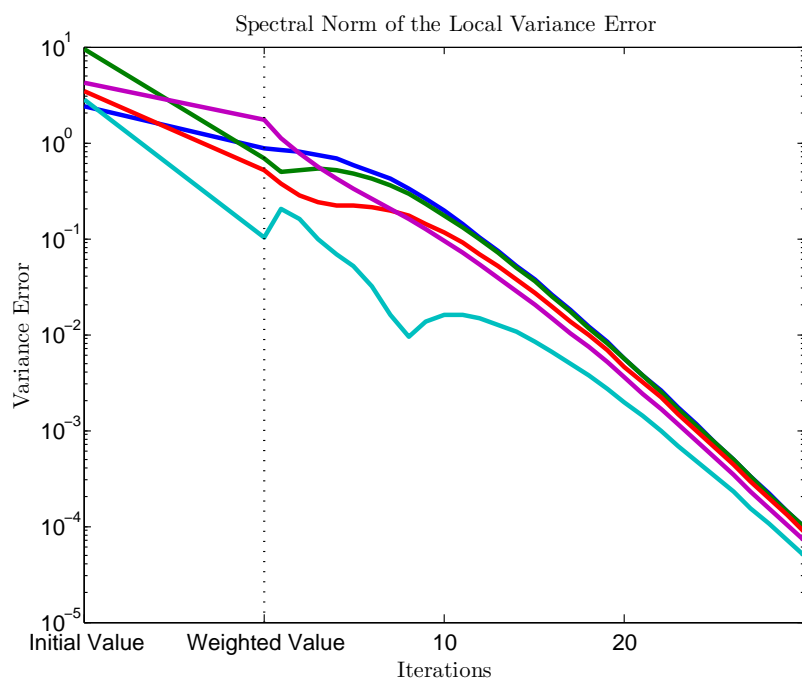


Figure 3-14: Local covariance error measured by  $\frac{\|\Sigma_{local} - \Sigma_{cent}\|_2}{\|\Sigma_{cent}\|_2}$  for the hyperparameter consensus on a known, unbalanced network

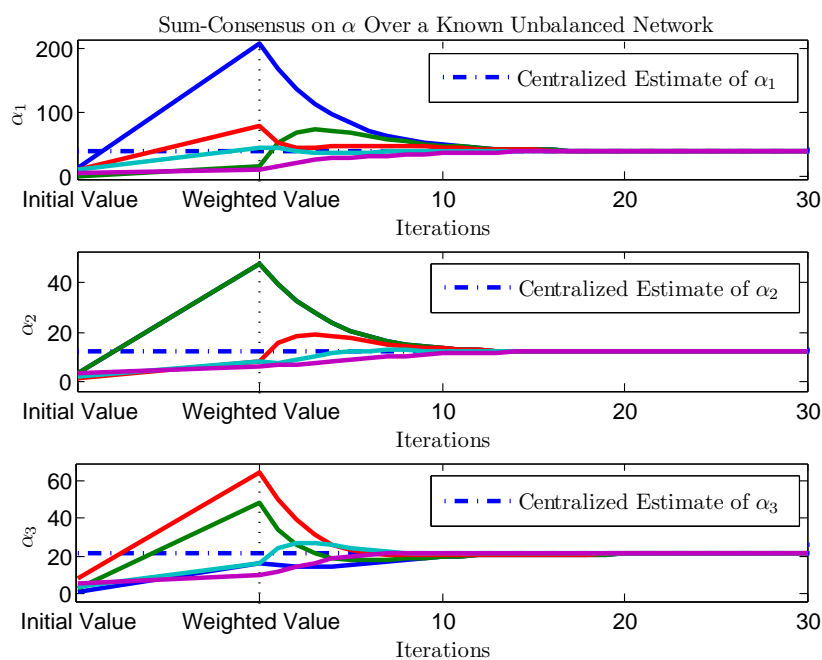


Figure 3-15: Hyperparameter trajectories during consensus on  $\mathbf{p}$  over a known, unbalanced network

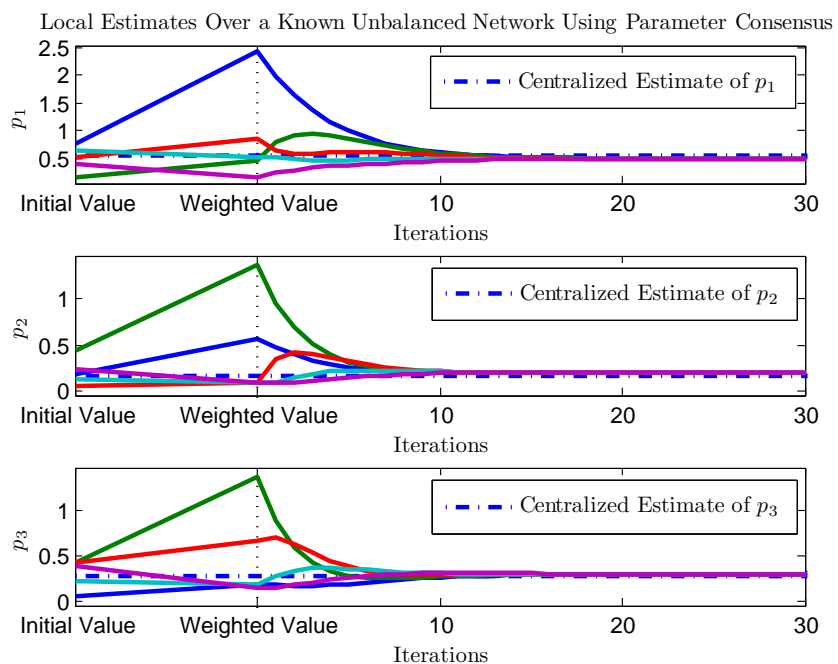


Figure 3-16: Local estimate achieved by the average consensus method on  $\mathbf{p}$  over a known, unbalanced network

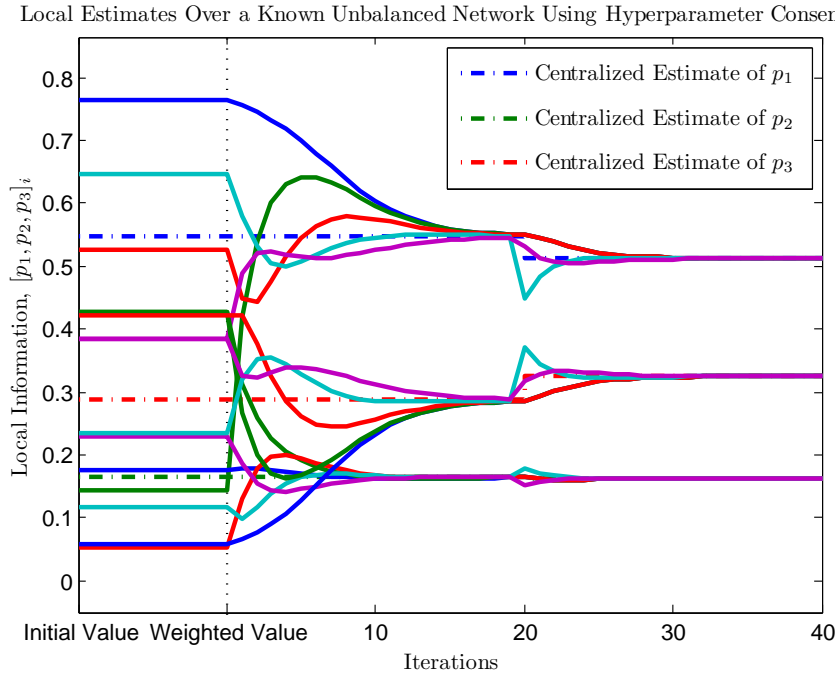


Figure 3-17: Local estimate achieved by hyperparameter consensus on  $\mathbf{p}$  with a measurement at  $t=20$

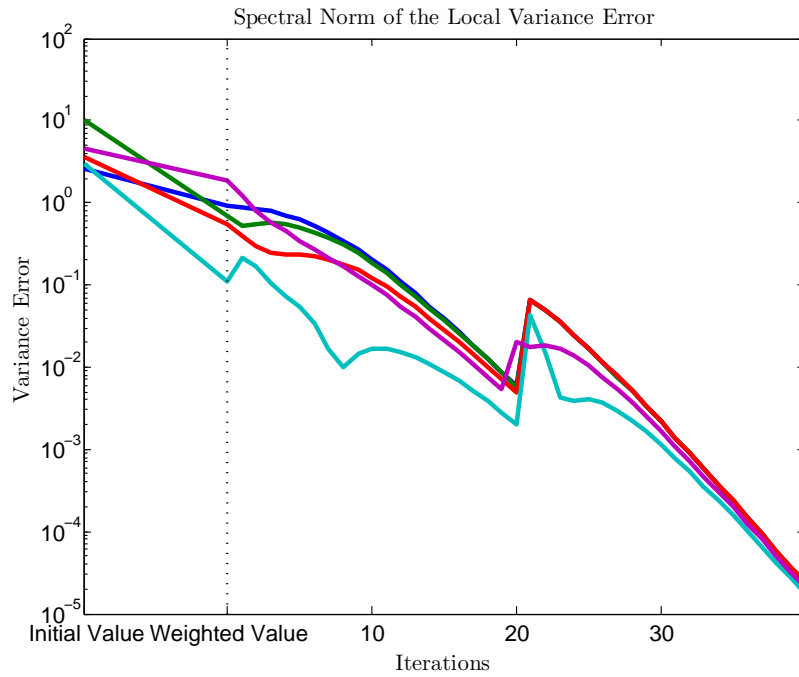


Figure 3-18: Local covariance error measured by  $\frac{\|\Sigma_{local} - \Sigma_{cent}\|_2}{\|\Sigma_{cent}\|_2}$  for the hyperparameter consensus known, unbalanced network with a measurement at  $t=20$

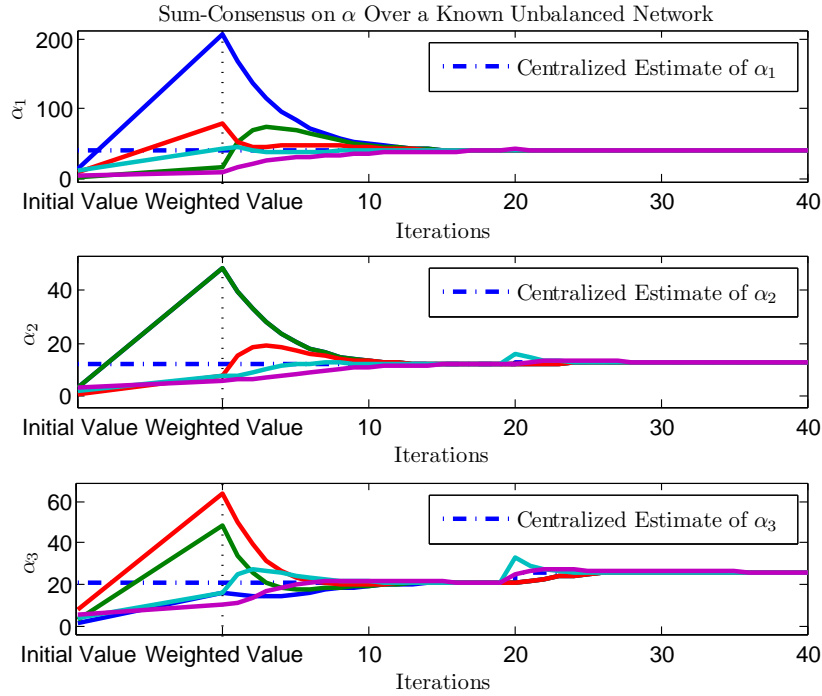


Figure 3-19: Hyperparameter trajectories during consensus on  $\mathbf{p}$  over a known, unbalanced network with a measurement at  $t=20$

### 3.3 Summary

This chapter introduced the distributed hyperparameter consensus method to allow multi-agent systems to converge to the proper centralized Bayesian parameter estimates. It began by re-introducing the Kalman filter as the optimal recursive Bayesian estimator of a normally distributed parameter for two purposes: 1) to motivate the formulation of uncertainties as parameterized probability distributions by an analogy with a well-known algorithm, and 2) to introduce the important concept of conjugate priors and hyperparameters. This Bayesian approach was extended to the subsequent derivation of the centralized Bayesian parameter estimate in the consensus framework and introduction of an improper consensus prior. Insights gained through the derivation of the centralized estimate were then used to create the hyperparameter consensus method which guarantees that local estimates will converge to the Bayesian estimate under standard network assumptions. The key innovation



of the hyperparameter consensus method is to run traditional consensus protocols on the hyperparameters rather than the parameters themselves. This shift of focus to the hyperparameters allows normal average consensus methods to implicitly transfer measures of uncertainty among agents and therefore bias the result properly towards those agents that are more confident.

This chapter also showed the ability of the hyperparameter consensus method to converge to a time-varying centralized estimate due to measurements made of a static process during consensus. This assumption of a static process is necessary to avoid temporal coupling of measurements and the need for a propagation step for the local information. If the estimated process is changing with time, then the distributed estimation problem may need to be separated into distinct epochs to allow for a propagation step of the local information according to some transition model (similar to the propagation step in the Kalman filter). Further complications arise due to the temporal coupling of agents' measurements and the fact that measurements need to be considered in proper order. An approximate heuristic is to apply a fading factor to the hyperparameters such that older measurements are given less weight than newer estimates, which will allow the estimates can track a slowly time-varying process. Though an interesting question, the problem of estimating a time-varying process was not explicitly considered in the scope of this thesis and is left as future work.

The hyperparameter consensus method was then illustrated through applications to the arrival rate and probability vector parameters through use of the gamma and Dirichlet conjugate priors, respectively. The gamma distribution was utilized as the conjugate distribution to the exponential and Poisson likelihood distributions, which are themselves parameterized by an arrival rate,  $\lambda$ . Using Bayesian principles, the hyperparameter consensus method was focused for the gamma distribution, and, in particular, consensus on the hyperparameters  $\alpha$  and  $\beta$ . It was shown that the hyperparameter consensus method allowed every agent to converge to the centralized parameter estimate over both balanced and unbalanced networks and in the presence of local measurements. Further, the hyperparameter consensus method was compared to parameter consensus on  $\lambda$  over an unknown, possibly unbalanced network. It

was not only shown that the hyperparameter consensus method expected to perform better for all values of network bias, but it was shown to also have a smaller variance in its error, suggesting that its worst case performance is also better than parameter consensus.

The Dirichlet distribution was introduced as a method to convey and communicate uncertainties in a probability vector,  $\mathbf{p}$ , during consensus. Due to the Dirichlet's limiting approximation as a normal distribution, the Kalman consensus algorithm was investigated in conjunction with the hyperparameter consensus. Despite the similarities, the Kalman consensus method was shown to converge to biased results even in the simplest case, with only the hyperparameter method converging on both balanced and unbalanced networks, with and without local measurements. Finally, it was also shown that consensus on the entries of the probability vector using traditional average consensus led to catastrophic transient behavior, especially in unbalanced networks, where the local probabilities did not remain within the unit simplex, thereby violating the fundamental axioms of probability theory.

# Chapter 4

## Application to Cooperative Markov Decision Processes

The hyperparameter consensus method introduced in the previous chapter not only allows agents to come to agreement with local uncertainties, but also facilitates the distributed estimation of an unknown parameter. By taking measurements using the same conjugate prior and measurement update as introduced for use in the consensus algorithm, agents can take measurements locally and subsequently combine their independent observations through agreement on the hyperparameters, effectively forming a distributed sensor network. In order to demonstrate this ability within a conventional framework, this chapter presents a simple but illustrative multiple-machine repair problem adapted from the single-machine repair problem in [7]. The problem will be used here to illustrate the sensitivity of resulting policies to knowledge of the underlying model during the learning process. In particular, multiple agents will be observing independent, identically distributed local machines and learning the state transition probabilities online, with the option of communicating with each other using either hyperparameter consensus or average consensus on the current model estimate.

This scenario can be applied to reliability analysis in the context of a group of new machines purchased by a company and distributed among a network of factories, in which each factory is concerned over its own local operations, but may commu-

nicate with other factories to share information in order to make more intelligent decisions. In a more aerospace-related framework, consider a number of reconnaissance teams outfitted with identical UAV resources and tasked with observing equally hostile environments. Each team has its own local goals to increase knowledge of the surroundings while maintaining consideration for the health of the local vehicle. By observing multiple missions, each team attempts to learn more of the underlying system model, and through communication with other teams, each can formulate a more accurate model upon which to base future actions.

The following chapter will introduce the basic Markov Decision Process (MDP) framework upon which the machine repair problem is based, and highlight the MDP's use in the reinforcement learning and estimation communities. This modeling structure will then be applied to the single-machine repair problem and, subsequently, extended to the multi-agent scenario. It is shown that agreement using hyperparameter consensus on the unknown model expedites learning in the large, cooperative multi-agent setting, and that this leads to significant performance improvement early in the estimation process.

## 4.1 Markov Decision Processes

Markov Decision Processes are a convenient modeling tool to describe how actions affect the state evolution of a process and the incurred rewards or penalties over time. All MDP representations maintain some set of states,  $s \in \mathbb{S}$ , and actions,  $a \in \mathbb{A}$ , as well as some method by which to encode the benefit of taking a certain action at a certain state,  $R(s, a) : \mathbb{S} \times \mathbb{A} \mapsto \mathfrak{R}$ , and a method to depict the state resulting from the action taken,  $T(s'|s, a) : \mathbb{S} \times \mathbb{S} \times \mathbb{A} \mapsto p \in [0, 1]$ . The transitions are often stochastic depending on the action taken at a certain state to depict the probabilities of arriving at a subsequent state. This chapter will consider the infinite horizon problem, in which the overall goal is to find a policy,  $\pi^*(s) : \mathbb{S} \mapsto \mathbb{A}$ , that maximizes

the expected time-discounted future given by:

$$\pi^*(s) = \arg \max_{\pi} E \left[ \sum_{t=0}^{\infty} \gamma^t R(x_t, \pi(x_t)) | x_0 = s \right]$$

where  $\gamma \in (0, 1)$  is a discount factor such that the infinite sum is finite in value, and the expectation is taken with respect to the stochastic transition probabilities. Though this chapter will focus on beneficial rewards, this approach can be similarly applied to minimize costs by switching from a maximization to a minimization.

Model-based representations explicitly represent the MDP as the collection of model parameters,  $M : \langle \mathbb{S}, \mathbb{A}, T, R \rangle$ , where the first two components are the aforementioned set of states and actions,  $T$  is a transition matrix that defines the probability of transitioning from state  $s$  to state  $s'$  by taking action  $a$ . For the remainder of the discussion, it will be assumed that  $T(s'|s, a)$  denotes the probability of reaching  $s'$  by taking  $a$  at  $s$ , and that  $\sum_{s'} T(s'|s, a) = 1 \forall s \in \mathbb{S}, a \in \mathbb{A}$ . The reward model to be used here is assumed to be deterministic and depends on either full or partial observation of the executed transition. Using a reward model over the full transition,  $R(s, s', a)$  (where  $s$  is the initial state,  $a$  is the action, and  $s'$  is final state), implies that the utility of an action is defined by the action as well as its outcome (for example, trying to walk across a tight-rope is only good if you make it to the other side). If the reward model  $R(s, s', a) = R(s, a)$ , then the execution of a particular action at a certain state has an associated utility, regardless of the outcome (i.e. a fuel-burn penalty associated with moving from a given state). Alternately, defining  $R(s, s', a) = R(s)$  or  $R(s')$  implies that any reward received is only dependent on the particular state itself, such as being in a goal or penalty state. In problems such as in the multi-arm bandit scenario, stochastic reward models may be used with similar relationships to the state/action dependencies[50].

Various solution methodologies exist for finding the optimal policies, including Dynamic Programming[55], Linear Programming, and Value Iteration. These rely on the use of the value function,  $V(s)$ , that encodes the long-term benefit of being in a

certain state.  $V(s)$  can be determined through the recursive Bellman equations:

$$\begin{aligned} Q_{t+1}(s, a) &= \sum_{s'} T(s'|s, a) [R(s, s', a) + \gamma V_t(s')] \\ V_{t+1}(s) &= \max_a Q_{t+1}(s, a) \end{aligned} \tag{4.1}$$

where it is assumed here that the rewards are given deterministically (the stochastic reward recursion is easily obtained) and  $V_0(s)$  is initialized to a finite value. The value function can also be denoted  $V^\pi(s)$ , which implies that the recursion in Eq. 4.1 occurs using actions determined according to a policy  $\pi(s)$ , such that  $a = \pi(s)$ , in lieu of the maximization. The optimal policy,  $\pi^*(s)$ , will select the most profitable action from any given state, assuming the same policy is also used to determine all future actions. It can be found as the policy that solves Bellman's optimality equation, and is given by

$$\pi^*(s) = \arg \max_a \sum_{s'} T(s'|s, a) [R(s, s', a) + \gamma V^{\pi^*}(s')]$$

### 4.1.1 Relationship to Reinforcement Learning

In many situations, it may not be possible to model the full MDP and solve for the optimal policy immediately. This problem could arise if, simply, the model is not known *a priori* or only coarse estimates of the model properties are available. If this is the case, it is required that the system be learned over time so that the optimal policy can, eventually, be obtained. This process of online learning is one of the fundamental components of reinforcement learning.

Reinforcement Learning (RL) methods utilize repeated interaction with the surroundings coupled with a received reward or penalty to determine whether the observed transition is beneficial or not. RL often uses a MDP framework to model the underlying environment, and seeks to learn a representation of this MDP based on current and past observations. Working with the representations listed in Section 4.1, RL can be approached through model-free or model-based techniques. Model-free methods learn the quality of state-action pairs through repeated execution and rewards [56, 57], while model-based techniques seek to learn the underlying model,

including transition probabilities and rewards, and use this to calculate the optimal policy [50, 51, 58]. The work in this section will focus on model-based learning and the estimation and sharing of transition probabilities.

## Model-Based RL

Most model-based RL techniques use Bayesian updates to keep a running estimate of the model of the world based on past observations. As the model is updated with new information the optimal policy is re-calculated to reflect the most recent knowledge.

Given that an agent's transition model is stochastic, such that the probability of arriving in state  $s'$  from state  $s$  given action  $a$  is  $T(s'|s, a)$ , we can generate a set of probability vectors  $\mathbf{p}_{s,a} = T(\cdot|s, a)$  such that  $\mathbf{p}_{s,a}^{s'} = T(s'|s, a)$ . For the moment, and without loss of generality, we will focus on a generic state-action pairing and drop the  $\langle s, a \rangle$  indexing, letting  $\mathbf{p}_{s,a}^{s'} \rightarrow \mathbf{p}_{s'}$ , and assume that the cardinality of the state space is  $L$ , such that  $|\mathbb{S}| = L$ , and, furthermore, denote these  $L$  states as being from the set  $\mathbb{S} := \{1, \dots, L\}$ . Therefore, the outcome will be state  $i$  with probability  $\mathbf{p}_i$ , with  $\sum_{i=1}^L \mathbf{p}_i = 1$ ,  $0 \leq \mathbf{p}_i \leq 1$ .

If we let a set of  $N$  observations be denoted  $Z^N = \{z_1, z_2, \dots, z_N\}$ , with each  $z_j \in \{1, \dots, L\}$ , then we can determine the probability of  $Z^N$  as:

$$Pr(Z^N | \mathbf{p}) = \prod_{i=1}^L \mathbf{p}_i^{N_i}$$

where  $N_i$  is the number of occurrences of the outcome  $i$  in the set  $Z^N$ . The previous probability is a multinomial distribution on the outcome of observations given a discrete probability distribution over the outcomes,  $\mathbf{p}$ . Thus, the Dirichlet distribution can be used to model the prior on the probabilities for each probability vector, and the probability of observing outcome  $i$  is

$$p(z = i | \alpha) = \int \mathbf{p}_i p(\mathbf{p} | \alpha) d\mathbf{p} = \frac{\alpha_i}{\sum_j \alpha_j}$$

These probabilities can be updated through observing the transitions and incre-

menting the count associated with the observed final state as in Section 3.2.3. This is the basis of Bayesian updates on the transition probabilities within the MDP framework. When agents are trying to learn an environment, they can maintain their experience history by updating the  $\alpha$  corresponding to the observed transition. As the number of observations grows, the approximate transition probabilities will approach the truth [47].

It is likely that not all states can be reached from every other state in a single transition, and so not all  $L$  outcomes will necessarily be observed from each state-action pair. Additionally, if it is wrongly assumed that a certain set of outcomes is reachable, it can often take many observations for the infeasible transitions' probabilities to become small (though, still, never zero), especially if the corresponding state-action pair occurs rarely. Friedman *et al.* formulate a hierarchical sparse Dirichlet representation that accounts explicitly for the unknown size of the set of reachable outcomes when determining the confidence in the local estimates[53]. This method relies on a prior distribution on the size of the reachable set and conditions the expected probabilities on the set size, updating both the probability distribution and set size distribution with each new update.

The MMSE estimate of the probability vector obtained through the sparse Dirichlet framework is equivalent to the MMSE probability vector obtained by a regular Dirichlet over the reduced set of outcomes,  $V$ , such that

$$\mathbf{p}_i = \begin{cases} \alpha_i/\alpha_0 & \text{if } i \in V \\ 0 & \text{o.w.} \end{cases}$$

This result can also be achieved through a slight abuse of the traditional Dirichlet by defining all  $\alpha_i = 0$  for  $i \notin V$  and using the usual MMSE parameter estimate equation for all probabilities:

$$\mathbf{p}_i = \frac{\alpha_i}{\sum_j \alpha_j}$$

By utilizing the Dirichlet prior on the distribution of transition probabilities and performing a Bayesian update based on observed transitions, agents can learn the



underlying transition model over time. Stochastic reinforcement models over a finite set of possible immediate rewards can similarly be learned using the Dirichlet prior[50, 51].

## 4.2 Machine Repair Problems

This section will use a simple machine repair problem to highlight the applicability of hyperparameter consensus to large-scale, cooperative learning problems. Before introducing the multi-machine problem, the single-machine problem is described. The multi-machine repair problem is then introduced as a cooperative, distributed estimation problem where multiple operators are attempting to learn the dynamics of a group of identical machines. Though the problem presented is fairly simple in order to obtain comprehensible results, the prescribed approach can be adapted to many multi-agent learning and estimation problems that fall within the MDP framework.

### 4.2.1 The Single-Machine Repair Problem

The single-machine repair problem that will serve as the basis for this chapter is a simple, two-state MDP in which, at time  $t$ , the machine is either broken ( $s_t = 0$ ) or working ( $s_t = 1$ ). If the machine is working, the operator receives \$100 and can choose to perform maintenance on the machine (action  $m$ ) which will improve the likelihood of the machine working at the next time step, but at a cost of  $C_m$  for parts and labor. The operator could, alternately, opt to do nothing at no additional cost (action  $n$ ), but with an increased risk of the machine breaking. If the machine is broken, the operator can either fix it (action  $f$ ) for a fee of  $C_f$  or replace it (action  $r$ ) for  $C_r$ , where it is constrained that, for any reasonable problem, the cost to replace is more than to fix. If the machine is replaced, then the new machine is guaranteed to work for the next stage. The model is outlined below:

$$\mathbb{S} = \begin{cases} \text{broken} : & s = 0 \\ \text{working} : & s = 1 \end{cases}$$

If the machine is working,  $s_t = 1$ :

$$\begin{aligned} \mathbb{A}(s_t = 1) &= \begin{cases} \text{maintenance : } & a_t = m \\ \text{nothing : } & a_t = n \end{cases} \\ R(s_t = 1, a_t) &= \begin{cases} 100 - C_m & \text{if } a_t = m \\ 100 & \text{if } a_t = n \end{cases} \\ p(s_{t+1}|s_t = 1, a_t = m) &= \begin{cases} 1 - \mathbf{p}_m & \text{for } s_{t+1} = 0 \\ \mathbf{p}_m & \text{for } s_{t+1} = 1 \end{cases} \\ p(s_{t+1}|s_t = 1, a_t = n) &= \begin{cases} 1 - \mathbf{p}_n & \text{for } s_{t+1} = 0 \\ \mathbf{p}_n & \text{for } s_{t+1} = 1 \end{cases} \end{aligned}$$

If the machine is broken,  $s_t = 0$ :

$$\begin{aligned} \mathbb{A}(s_t = 0) &= \begin{cases} \text{fix : } & a_t = f \\ \text{replace : } & a_t = r \end{cases} \\ R(s_t = 0, a_t) &= \begin{cases} -C_f & \text{if } a_t = f \\ -C_r & \text{if } a_t = r \end{cases} \\ p(s_{t+1}|s_t = 0, a_t = f) &= \begin{cases} 1 - \mathbf{p}_f & \text{for } s_{t+1} = 0 \\ \mathbf{p}_f & \text{for } s_{t+1} = 1 \end{cases} \\ p(s_{t+1}|s_t = 0, a_t = r) &= \begin{cases} 0 & \text{for } s_{t+1} = 0 \\ 1 & \text{for } s_{t+1} = 1 \end{cases} \end{aligned}$$

The problem considered here assumes that the operator knows the costs associated with each state action pair (ie.  $R$  is fully known), but the transitions  $T(s_{t+1}|s_t, a_t)$  are unknown and must be learned. When the machine is working, the transition matrix is parameterized by the probabilities  $\mathbf{p}_m$  and  $\mathbf{p}_n$ , which designate the probability of the machine working at the next time step after, respectively, performing maintenance,

$m$ , or taking no action,  $n$ :

$$T(\cdot|s_t = 1, a_t) = \begin{bmatrix} 1 - \mathbf{p}_m & \mathbf{p}_m \\ 1 - \mathbf{p}_n & \mathbf{p}_n \end{bmatrix} = \begin{bmatrix} \mathbf{p}_m \\ \mathbf{p}_n \end{bmatrix}$$

When the machine is broken, it is assumed that replacing the machine is known to guarantee a working state at the next stage, so the transition matrix is parameterized only by the probability of the machine working given having fixed it,  $\mathbf{p}_f$ :

$$T(\cdot|s_t = 0, a_t) = \begin{bmatrix} 1 - \mathbf{p}_f & \mathbf{p}_f \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{p}_f \\ \mathbf{p}_r \end{bmatrix}$$

The objective is to determine an optimal policy for each machine state,  $\pi^*(s)$ , that maximizes the time-discounted future reward

$$\pi^*(s) = \arg \max_{\pi} V^{\pi}(s) = \arg \max_{\pi} E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, \pi(s_t)) \middle| s_0 = s \right]$$

where the expectation is taken over the state transitions,  $T(s_{t+1}|s_t, \pi(s_t))$ . This problem can be solved using value iteration, where the recursion in Eq. 4.1 becomes

$$V_{i+1}(s) = \max_{a \in \mathbb{A}(s)} R(s, a) + \gamma \sum_{s' \in \mathbb{S}} T(s'|s, a) V_i(s')$$

where  $V_0(s)$  is initialized arbitrarily and the recursion is run until convergence. The simulations presented in this chapter use a time discount factor  $\gamma = 0.8$ .

Uncertainty in the probability vectors  $\mathbf{p}_a = [1 - \mathbf{p}_a \ \mathbf{p}_a]$  can be described by a beta distribution, the univariate form of the Dirichlet. The beta distribution for action  $a$  is parameterized by counts  $\alpha_a$  and  $\beta_a$ , which represent the number of times action  $a$

has resulted in a working or broken state, respectively, and are updated by:

$$\alpha_a(t+1) = \begin{cases} \alpha_a(t) + 1 & \text{if working at } t+1 \\ \alpha_a(t) & \text{if broken at } t+1 \end{cases}$$

$$\beta_a(t+1) = \begin{cases} \beta_a(t) & \text{if working at } t+1 \\ \beta_a(t) + 1 & \text{if broken at } t+1 \end{cases}$$

when action  $a$  is taken. As an operator explores the state-action space, the counts are updated to reflect the observed transitions and are used to determine new probabilities for the solver. An undirected Boltzmann method will be used to capture the exploration vs exploitation trade-off. This approach utilizes the currently expected Q-values to determine the probability of selecting each action according to

$$Pr(a|s) = \frac{e^{Q(s,a)/T}}{\sum_{a'} e^{Q(s,a')/T}}$$

where  $T$  is a heuristic “temperature” parameter that starts large (so all actions are equally likely) and is set to decay over time to promote greedy selection in the long-run. In the following simulations the temperature parameter has been tuned to a large value with a slow decay rate. This promotes a longer exploration phase, such that the compared methods take approximately equally random actions regardless of the knowledge of the model and the resulting performance is based on the consensus algorithm rather than the exploration one.

### 4.2.2 The Multi-Agent Machine Repair Problem

In the multi-agent setting,  $N$  machines are run independently by  $N$  agents, but the agents are allowed to share information among themselves to try and agree upon the model of the system. Since each agent observes an independent machine, the measurements it makes are independent from the state or measurements of any of the other machines. For comparison, the hyperparameter consensus algorithm and the average consensus algorithm were used, where the average consensus was run on

the local probability estimates themselves. The results shown here are for a range of network sizes where agents run each of the consensus algorithms to convergence at various points in the estimation process and evaluate the resulting agreed-upon model. Thus, this comparison shows the relative value of consensus at varying stages of the learning process. Unfortunately, the comparison cannot be easily extended to concurrent estimation and consensus problems. This is because the estimation problem utilizes the Dirichlet counts, which are preserved by the hyperparameter consensus method but are not uniquely defined after executing average consensus on the probabilities themselves<sup>1</sup>. Therefore, not only do the following comparisons show the benefit of using the hyperparameter consensus from a performance standpoint, but through maintaining representative counts, hyperparameter consensus is able to work concurrently with the Bayesian estimation scheme where other approaches (Kalman Consensus and Average Consensus) cannot. Finally, it is necessary to note that the results shown are obtained only after the agents have agreed on the model parameters. This assumption allows the results to be generalized to an arbitrary, known, strongly connected network such that the desired hyperparameter consensus result can be obtained, and so that inconsistencies in the probabilities that occur during the transient of the average consensus algorithm (as shown in Section 3.2.3) are ignored and only the resulting steady-state values (which will lie in the unit simplex) are used.

Each local learning problem was simulated with the true model as given below. If the machine is working at a given time, the cost of maintenance is  $C_m = \$10$  and

---

<sup>1</sup>Since probabilities alone are not enough to uniquely define the counts, additional information, like the variance, is required but unavailable when using average consensus. Further, though the mean and variance will uniquely define the counts, in Section 3.2.3 the mean-variance approach using a Kalman Consensus Filter was shown to provide biased results, and is not considered here for that reason

raises the likelihood of the continued functionality of the machine from 0.5 to 0.9:

$$\begin{aligned}
 R(s_t = 1, a_t) &= \begin{cases} 90 & \text{if } a_t = m \\ 100 & \text{if } a_t = n \end{cases} \\
 p(s_{t+1}|s_t = 1, a_t = m) &= \begin{cases} 0.1 & \text{for } s_{t+1} = 0 \\ 0.9 & \text{for } s_{t+1} = 1 \end{cases} \\
 p(s_{t+1}|s_t = 1, a_t = n) &= \begin{cases} 0.5 & \text{for } s_{t+1} = 0 \\ 0.5 & \text{for } s_{t+1} = 1 \end{cases}
 \end{aligned}$$

If the machine is broken, however, the operator can either spend \$70 to fix the machine or \$110 to replace it. Since the machine had been previously broken, extra stress may have fatigued other parts, such that fixing it leads to a working machine at the next step only 80% of the time.

$$\begin{aligned}
 R(s_t = 0, a_t) &= \begin{cases} -70 & \text{if } a_t = f \\ -110 & \text{if } a_t = r \end{cases} \\
 p(s_{t+1}|s_t = 0, a_t = f) &= \begin{cases} 0.2 & \text{for } s_{t+1} = 0 \\ 0.8 & \text{for } s_{t+1} = 1 \end{cases} \\
 p(s_{t+1}|s_t = 0, a_t = r) &= \begin{cases} 0 & \text{for } s_{t+1} = 0 \\ 1 & \text{for } s_{t+1} = 1 \end{cases}
 \end{aligned}$$

The results that follow are the outcome of 300 Monte-Carlo simulations to determine the average response given the true model and some set initial conditions. Three primary metrics will be used to judge the quality of the learning:

- Probability of finding the optimal policy: This metric determines the likelihood of obtaining the optimal policy through value iteration using the best estimate of the transition probabilities. As more measurements are made, the transition probabilities will converge to the truth and will result in convergence to the optimal policy. This metric is, however, sensitive to the *basin of attraction* of the optimal policy, as it may be possible to obtain the optimal policy even with

an incorrect model as will be shown at the end of this section.

- Expected error in the discounted future reward: This metric describes the difference between the average future reward expected using the current model estimate and the true value of the state under the optimal policy. Like the probability of optimal policy metric, the estimated value function is expected to converge to the truth with infinite measurements, but is a smoother, continuous metric that is an overall measure of how well the MDP model has been learned.
- Error in model parameters: In addition to MDP performance metrics, estimation performance will be evaluated using the expected values of the parameters themselves.

Using a uniform initial prior on the counts, such that  $\alpha_a = \beta_a = 1, \forall a \neq r$  (since replacing is a deterministic transition), and assuming that this prior information is common among the agents, the multi-agent machine repair problem was simulated for a range of agents from  $N = 2$  to  $N = 200$ . To account for the shared initial prior information, two methods for storing, updating, and sharing hyperparameters can be used:

- Each agent maintains a local estimate of the probabilities achieved through normal hyperparameter updates (Eq. 3.8) as well as knowledge of any shared, global information. The hyperparameter consensus method can be derived using Equation 3.6 and following the derivation in Section 3.1.2 with a proper prior equivalent to all shared information. Each agent can then run the hyperparameter consensus algorithm on the difference between the current local hyperparameters and the shared hyperparameters, and then add the result back to the shared hyperparameter value after convergence.<sup>2</sup>

---

<sup>2</sup>Note that this approach is similar to channel filtering in the data fusion community [34], except for one key difference: each agent needs to only keep track of two sets of hyperparameters (its own, local hyperparameters and the shared, global hyperparameters) instead of maintaining shared information between it and all of its neighbors as is typically required of channel filters.

- Each agent can update their prior with weighted measurements as in Equation 3.17, leaving the prior un-weighted (i.e. do not multiply the prior by  $1/\nu_i$ ). During consensus, this has the effect of running average consensus on the prior and sum consensus on the measurements, such that the prior remains counted only once but the measurements are given the desired weight.

In situations where there are a large number of measurements that separate complete consensus executions (such as when agents take measurements without communication, stop taking measurements and run consensus to convergence, then continue taking measurements), the first approach is likely desirable. Since the estimation and agreement problems are effectively separated, the first method allows each agent to maintain a local estimate that is not impacted by the consensus weights ( $1/\nu_i$ ) that are applied for hyperparameter consensus. If the consensus epochs are well defined in the sense that every agent knows when to start communicating and when convergence is reached, then the problem of maintaining knowledge of the shared information is not difficult since it is simply the result of the most recent consensus. If it is desired that consensus and estimation occur simultaneously, the second approach is beneficial as it requires no distinction between an estimation epoch and a consensus epoch as either a measurement can be made or messages sent or received at any time without any additional processing (such as subtracting and adding shared information in the previous method). Since the comparison problem to be addressed here fits into the first framework, this method will be used for the following results.

## Simulation Results

Using the hyperparameter consensus algorithm, the agents are able to utilize every measurement made by each agent and are able to learn the model much faster than using average consensus on the probabilities, as shown in Figure 4-1. Since all agents have an identical prior, both methods start with the same initial estimate. However, using the hyperparameters allows the agents to effectively combine their observations, such that 100 agents with 2 independent measurements each have the estimation power of a single agent with 200 measurements. Using average consensus



on the probabilities themselves, it appears that the probability estimates follow the same profile regardless of the number of agents, suggesting that there is little or no benefit to the estimation problem from agreement through averaging of the probabilities. Further, recall that the use of the linear consensus method for combining the hyperparameters implicitly accounts for the network topology, so there is no need for channel filtering on complex networks as would be required to obtain the same result using traditional distributed data fusion techniques.

This expedited learning ability allows the agents running the hyperparameter consensus to obtain the optimal policy more often and after fewer local measurements than parameter consensus, as shown in Figure 4-2. The top plot shows benefit of using hyperparameter consensus early in the estimation problem where its more accurate model leads to the agents finding the true optimal policy with probability approaching unity, while post-parameter consensus models almost always achieve a sub-optimal policy. This “transient” period (in which, traditionally, agents have not fully learned the true model) is where it is important to use hyperparameter consensus since it can take advantage of the combined measurements of all the agents. However, after sufficient local measurements have been made, each agent will individually converge to the true model, so that, in the limit, the two methods will give equivalent performance results.

Figures 4-3 and 4-4 show the expected evolution of the discounted future reward given the best estimate of the model. As with the other results, initially and in the limit of infinite measurements the two approaches give the same results. In the transient period, however, the hyperparameter consensus method converges much faster to the true future reward. The top plot in both figures shows the difference between the errors in two estimates obtained, and highlights the fact that the hyperparameter estimate is closer to the true value than parameter consensus by over 50% of the true value.

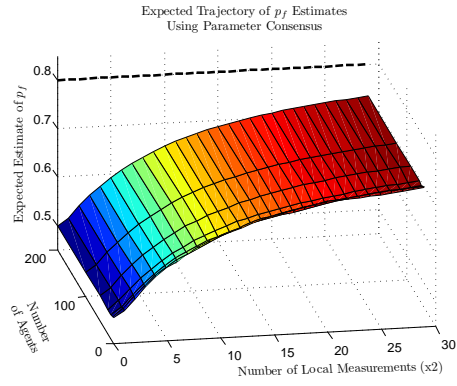
As an interesting note, the reduced dimensionality of the problem space allows for a unique visualization of the optimal policies obtained for different transition models, parameterized by  $p_f$ ,  $p_m$ , and  $p_n$ . Figure 4-5 shows a representation of the basins

of each of the four possible policies ( $\pi^* \in \{[f, m], [f, n], [r, m], [r, n]\}$ ) such that a set of values for the three uncertain probabilities is considered within a policy's basin if the probabilities define a model that would result in that particular policy being deemed optimal. For example, the point  $[p_f \ p_m \ p_n] = [0.5 \ 0.5 \ 0.5]$ , which is the initial probability estimate of all agents and is denoted by a black triangle in the figure, falls within the yellow basin corresponding to the policy  $\pi^* = [r, n]$ , meaning replace if broken and do nothing if working, whereas the true model probabilities of  $[0.8 \ 0.9 \ 0.5]$ , shown as the green circle, lead to a policy of fixing if broken and performing maintenance if working. The two curves plotted in the figure denote the post-consensus parameter estimates for both the hyperparameter consensus and average consensus on the probabilities for  $N = 200$ . The hyperparameter consensus trajectory in blue immediately jumps from the initial conditions to within the true optimal policy basin in dark blue, and, in fact, to a point very close to the true parameters as was suggested by Figure 4-1. The red curve denotes the average consensus parameter estimates, which actually traverse through the yellow and red basins corresponding to sub-optimal policies before, eventually, entering the true optimal policy basin near the end of the simulation.

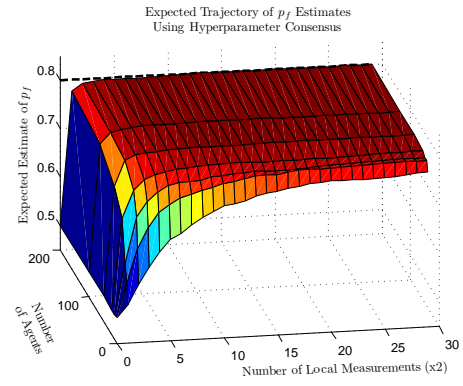
Similar results can be obtained using more informative priors or unique local priors that aren't shared across all agents. In the first case, the results are the same as those shown here but effectively started after taking some prior measurements, which would look like the tail end of the figures and provides the same benefits as already described. The second extension has the same traits, but may not directly mimic the results shown here since each agent is allowed to have a different initial condition. It is important, however, for the initial conditions to still be representative, or else they will bias the results when using hyperparameter consensus, which will not be able to distinguish between the uninformed and informed hyperparameters. If the initial conditions are uncertain and could be biased, then it may be beneficial to add a fading factor to the measurements (akin to process noise in the Kalman Filter setting) such that old information is given less weight than newer information.

Finally, in this section and the next, no explicit mention of the network topology

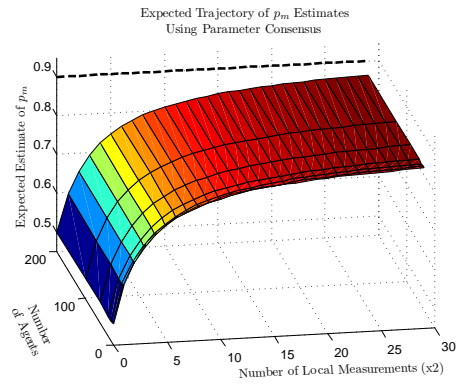
is made. This is done for two reasons: first, given each agent's current local estimate, the models used for the results considered here are the outcome of different consensus methods on these models, which have been run to a steady-state value and, thus, avoid the consensus transient period in which the average consensus estimates would be heavily impacted by network structure (eg. see Figure 3-15). By taking this approach, the displayed results are independent of the inherent bias in the network. Second, both average consensus and hyperparameter consensus have the same convergence properties over a broad range of networks, such that the only limitation to being able to obtain the demonstrated results are that the network be strongly connected and the consensus eigenvector be known. Again, this is in contrast to typical data fusion approaches where, to obtain the same results as the hyperparameter consensus algorithm, complex channel filters would be required which are heavily dependent on the network topology.



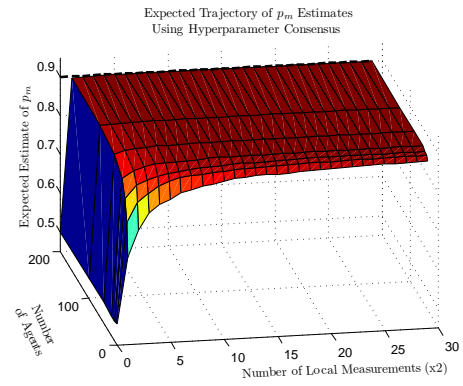
(a) Average consensus on  $p_f$



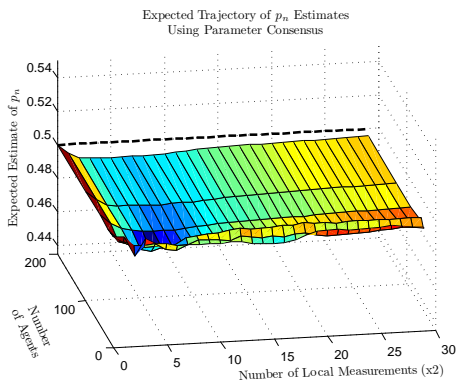
(b) Hyperparameter consensus on  $p_f$



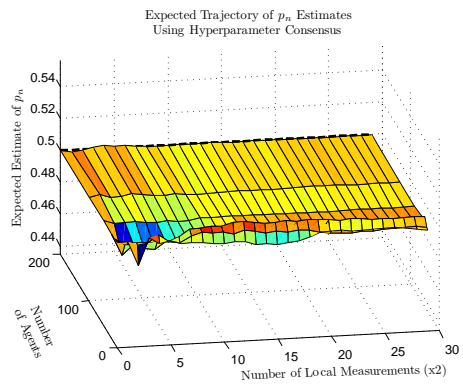
(c) Average consensus on  $p_m$



(d) Hyperparameter consensus on  $p_m$

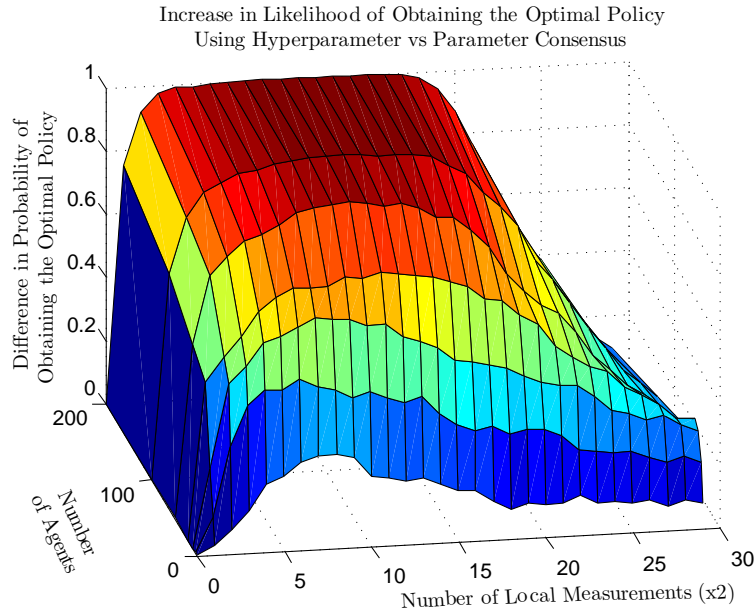


(e) Average consensus on  $p_n$

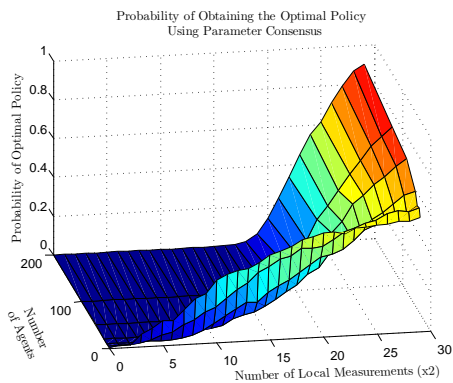


(f) Hyperparameter consensus on  $p_n$

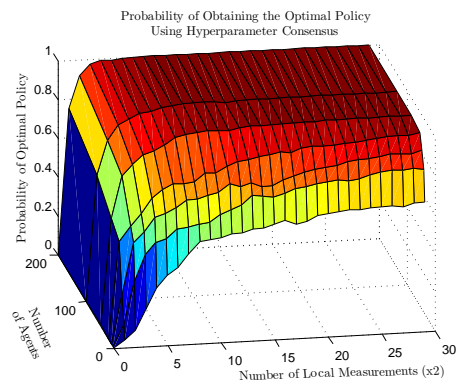
Figure 4-1: Expected probability estimates using each consensus method for varying number of local measurements



(a) Increased probability of obtaining the optimal policy using hyperparameter consensus over average consensus on the probabilities

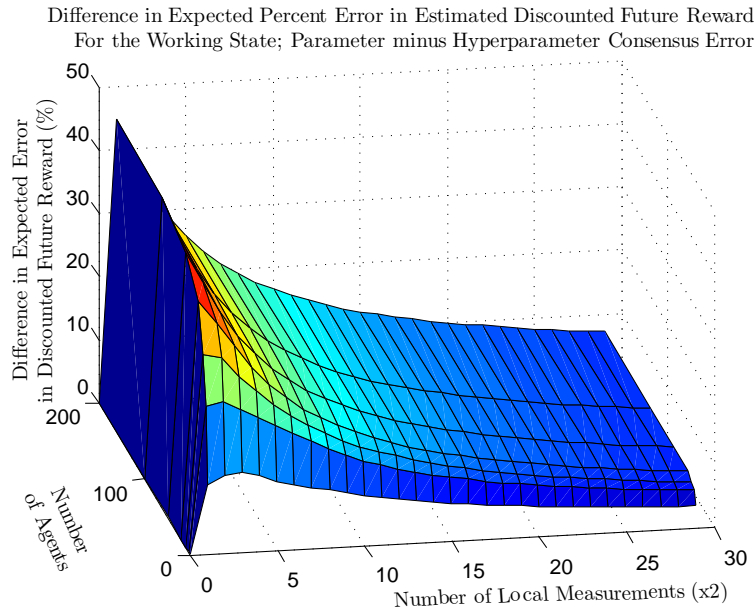


(b) Average consensus

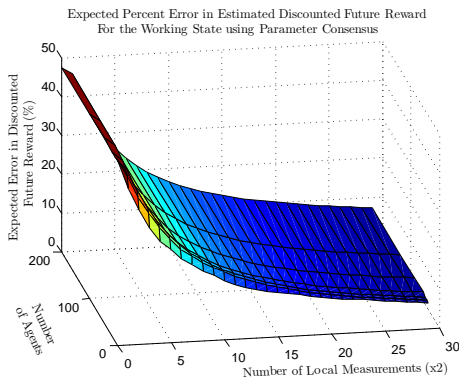


(c) Hyperparameter consensus

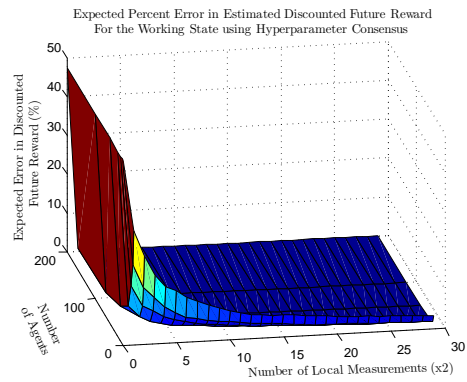
Figure 4-2: Probability of obtaining the optimal policy using each consensus method for varying number of local measurements



(a) Difference in expected error in the discounted future reward for the working state using hyperparameter consensus over average consensus on the probabilities

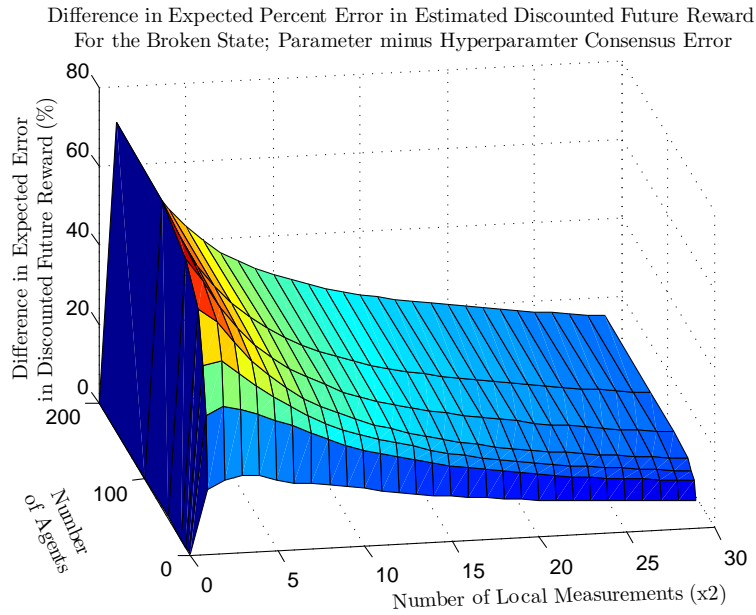


(b) Average consensus

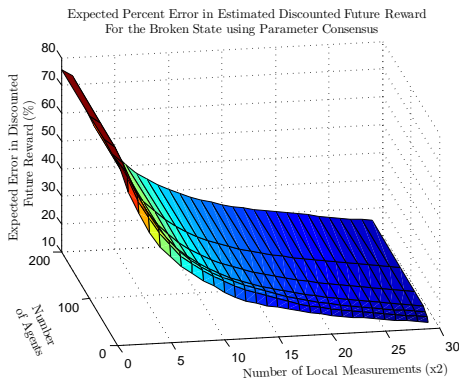


(c) Hyperparameter consensus

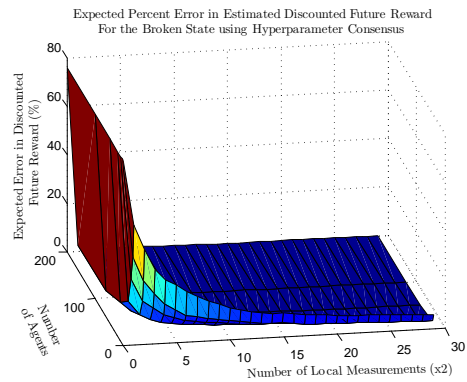
Figure 4-3: Expected error in the discounted future reward for the working state using each consensus method for varying number of local measurements



(a) Difference in expected error in the discounted future reward for the broken state using hyperparameter consensus over average consensus on the probabilities



(b) Average consensus



(c) Hyperparameter consensus

Figure 4-4: Expected error in the discounted future reward for the broken state using each consensus method for varying number of local measurements

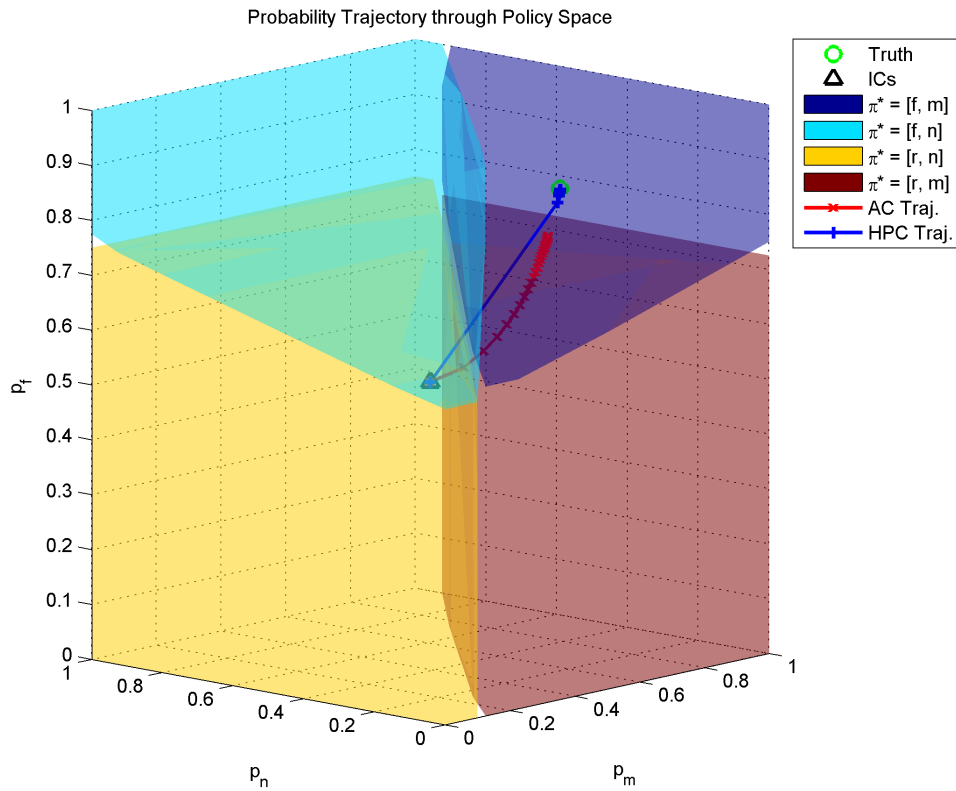


Figure 4-5: Trajectories of the probability estimates through probability space, superimposed over policy basins, for  $N = 200$ . Hyperparameter consensus (blue curve) estimates converge immediately to the true optimal policy,  $[f, m]$  (dark blue basin), while average consensus on the probabilities (red curve) travels through the  $[r, m]$  (yellow basin) and  $[r, n]$  (red basin) policies before eventually ending up in the optimal.



### 4.2.3 The Multi-Agent Machine Repair Problem with Time Considerations

This section considers a slightly more complicated model than previously presented whereby each action has an associated time required to complete it before the next action can be taken. For example, the maintenance action requires hiring a mechanic and performing the required maintenance in addition to the time required to fulfill the machine's normal duties, and is likely to take longer than not performing maintenance. However, these times are not always the same (some repairs may be harder than others), and so the time required for each action is determined to be an exponentially distributed random variable,  $\tau(a) \sim f_E(\tau(a)|\lambda_a)$ . Each action has an average time to complete, found as

$$E[\tau(a)] = 1/\lambda_a$$

In order to reflect this added temporal aspect, the expected future discounted reward is reformulated as

$$E_{T(s'|s,a),\tau(a)} \left[ \sum_{k=0}^{\infty} \gamma^{t_k} R(x_k, \pi(x_k)) \middle| x_0 = s, t_0 = 0 \right]$$

where

$$t_{k+1} = \sum_{i=0}^k \tau(a_i)$$

and the expected value is now over the time to complete each action as well as the state transitions. The corresponding optimal policy is

$$\pi^*(s) = \arg \max_{\pi} E_{T(s'|s,a),\tau(a)} \left[ \sum_{k=0}^{\infty} \gamma^{t_k} R(x_k, \pi(x_k)) \middle| x_0 = s, t_0 = 0 \right]$$

As with the previous model, the optimal policy and future reward can be found as

$$\pi^*(s) = \arg \max_a Q_{ss}(s, a)$$

$$V^*(s) = \max_a Q_{ss}(s, a)$$

where  $Q_{ss}(s, a)$  is the steady-state solution to the modified recursion,

$$\begin{aligned} Q_{t+1}(s, a) &= E_{T(s'|s, a), \tau(a)} [R(s, s', a) + \gamma^{\tau(a)} V_t(s')] \\ &= R(s, a) + E [\gamma^{\tau(a)}] \sum_{s'} T(s'|s, a) V_t(s') \\ V_{t+1}(s) &= \max_a Q_{t+1}(s, a) \end{aligned}$$

and where

$$E [\gamma^{\tau(a)}] = \int_0^\infty \gamma^t \lambda_a e^{-\lambda_a t} dt = \frac{\lambda_a}{\lambda_a - \ln(\gamma)}$$

Thus, the inclusion of a temporal aspect for each action has the effect of changing the discount value for any future rewards. Note that the proper discounting trend is maintained, such that actions with a long expected time-to-complete, corresponding to a small  $\lambda$ , will discount the future rewards heavily, while as  $\lambda \rightarrow \infty$ , the actions become instantaneous and the discount factor approaches unity. Since the discount is bounded by  $[0, 1)$  and  $T$  and  $V(s)$  are finite, the recursion converges to a steady state, finite  $Q_{ss}(s, a)$  and the expected discounted future reward also exists and is finite.

As was introduced in Section 3.2, the gamma prior is conjugate to the exponential distribution and will be used to represent the local uncertainty in the estimate of the rate parameters,  $\lambda_a$ . To avoid confusion with the  $\alpha_a$  and  $\beta_a$  already used for the beta distributions, the gamma will be described by hyperparameters  $A_a$  and  $B_a$ . As explained in Section 3.2.1, the gamma hyperparameters are updated after executing action  $a$  as:

$$A_a(t_{k+1}) = A_a(t_k) + 1, \quad B_a(t_{k+1}) = B_a(t_k) + (t_{k+1} - t_k)$$

The plots in Figures 4-6 to 4-12 depict the results from 300 Monte-Carlo simulations where the true values of  $\lambda_a$  and corresponding expected time to complete each

action,  $E[\tau(a)]$ , are given by

$$\begin{aligned}
 \lambda_f &= 0.2 & E[\tau(f)] &= 5 \\
 \lambda_r &= 0.2 & E[\tau(r)] &= 5 \\
 \lambda_m &= 0.5 & E[\tau(m)] &= 2 \\
 \lambda_n &= 1 & E[\tau(n)] &= 1
 \end{aligned}$$

The Dirichlet priors on the probabilities were initialized to the same “flat” distribution as used in the previous section, while the gamma hyperparameters were initialized as

$$A_a^{init} = 1, \quad B_a^{init} = 2$$

to give an initial  $\lambda_a$  estimate of 0.5. Each agent’s gamma prior on  $\lambda_a$  was then initialized to these same, common prior hyperparameters for each action ( $A_a^n = A_a^{init}$ ,  $B_a^n = B_a^{init} \forall n = [1, \dots, N]$ ).

## Simulation Results

The probability and rate parameter estimation problems are decoupled for any given action, which leaves the probability model estimates in Figure 4-6 unaffected by the inclusion of the temporal aspect and displaying the same quick convergence using the hyperparameter consensus method as in the previous section. Similarly, by aggregating the temporal observations through summation of the hyperparameters, the agents are also able to come to a better estimate of the  $\lambda$  values quicker than when using average consensus, as shown in Figures 4-7 and 4-8.

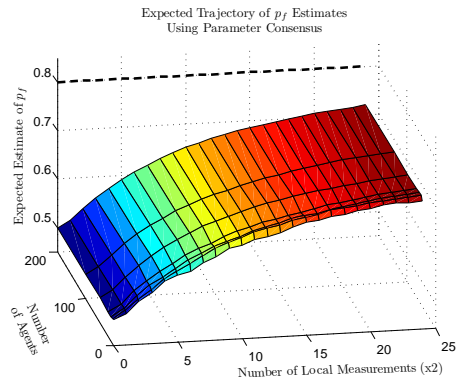
While the probability estimates were not affected by the temporal aspect, the inclusion of times does affect the performance by, in this instance, aiding the post-parameter consensus model in obtaining the optimal policy more often and earlier than in the previous section, as shown in Figures 4-9 to 4-11 (this result will be revisited and explained in a moment when discussing the basins of attraction of the various policies). The probability of obtaining an optimal policy is, again, increased in the early-going using hyperparameter consensus, such that parameter consensus

only begins to match the hyperparameter consensus likelihoods after 30 to 40 local measurements, and, even then, only in the large-scale, 200 agent scenario. This, again, shows the benefit of the quick convergence to model parameters early in the estimation problem, even though this problem leads to a shorter transient period than in the previous section. The estimated future value for the working state in Figure 4-10 shows much the same trend as in the previous section, where it is able to converge much faster to the true distribution. Specifically, the hyperparameter consensus method converges to within 5% of the true discounted future reward after less than 10 local measurements by each agent (faster for more agents), while it takes the parameter consensus method upwards of 50 local measurements to converge within comparable error.

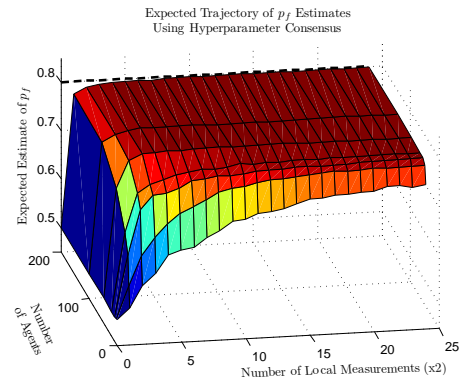
The expected discounted future reward for the broken state, shown in Figure 4-11, displays an interesting result due to a confluence of estimation values during the parameter consensus. Initially, both methods underestimate the true discounted reward, though the hyperparameter consensus results are in keeping with previous results and converge over large networks to within 5% of the true value after 2 local measurements, and take a little longer for convergence in the smaller networks of 2 to 5 agents. The average consensus method, however, ends up converging to and subsequently overshooting the true expected value around 10 local measurements due primarily to the over-estimation of  $\lambda_f$  and  $\lambda_r$ . This over-estimation due to the initial conditions causes the agents to expect these actions to be accomplished quickly, and therefore discounts the expected benefit of the subsequent working state less than it should, causing the expected future reward to quickly increase to and mildly surpass the true value, though not because of a good knowledge of the model. Despite this serendipitous result, the difference in expected errors in the discounted future reward peaks at over 100% of the true value, which is a huge error that the hyperparameter consensus method is able to avoid. The negative result is due to non-zero error in the hyperparameter consensus results at the point where the parameter consensus estimate is passing through the true value.

As alluded to in the preceding discussion, the  $\lambda$  estimates can play a large role in

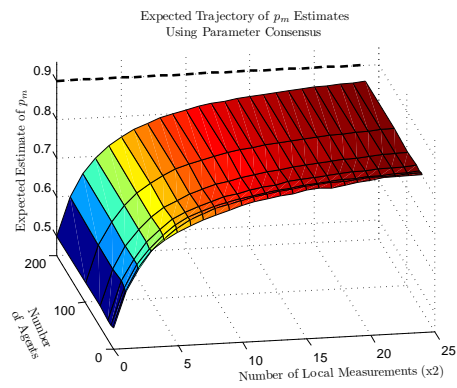
the solution of the MDP. Figure 4-12 shows the same three-dimensional visualization of the policy basins for two sets of estimates of  $\lambda_a$ : the initial conditions,  $\lambda_a = 0.5$ , on the top, and using the true  $\lambda_a$  on the bottom. Again, the probability estimate trajectories are plotted for  $N = 200$ , starting at the initial conditions of  $\mathbf{p}_a = 0.5$ . The impact of the temporal properties on the basins is quite noticeable as the volumes associated with fixing the machine when broken have both grown. It follows from this sensitivity to the temporal characteristics of the problem that the proper estimation of the execution times is critical for properly estimating the response of the system. This also explains why the average consensus method was able to obtain a higher likelihood of optimal policies earlier than in the previous problem despite an equivalent probability estimate trajectory: the policy basins for the estimated  $\lambda$  trajectories (which form a series of configurations between the two shown) evolve such that the basin of the optimal policy (dark blue) grows and encompasses the estimated probability trajectory earlier in the estimation process than happened in the non-temporal example.



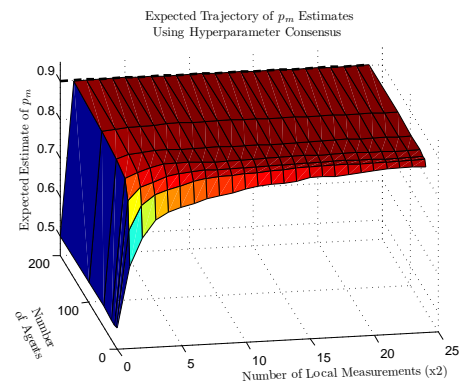
(a) Average consensus on  $\mathbf{p}_f$



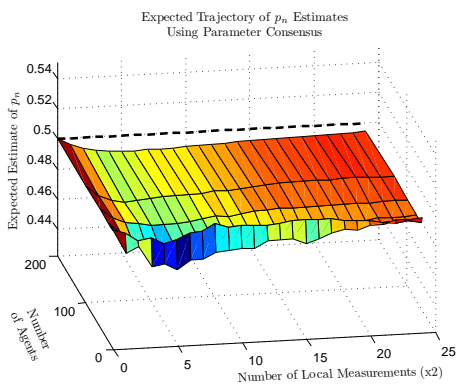
(b) Hyperparameter consensus on  $\mathbf{p}_f$



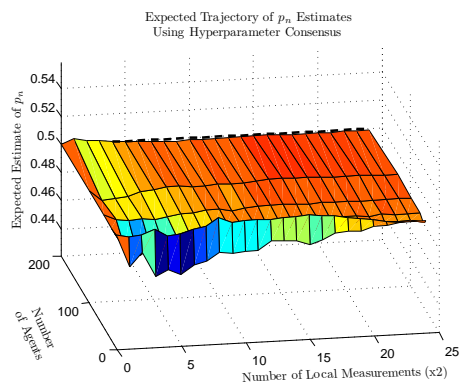
(c) Average consensus on  $\mathbf{p}_m$



(d) Hyperparameter consensus on  $\mathbf{p}_m$

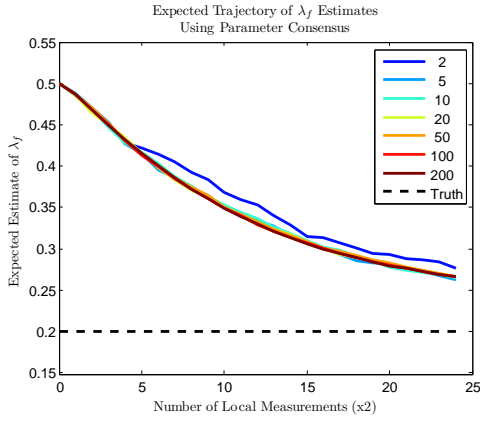


(e) Average consensus on  $\mathbf{p}_n$

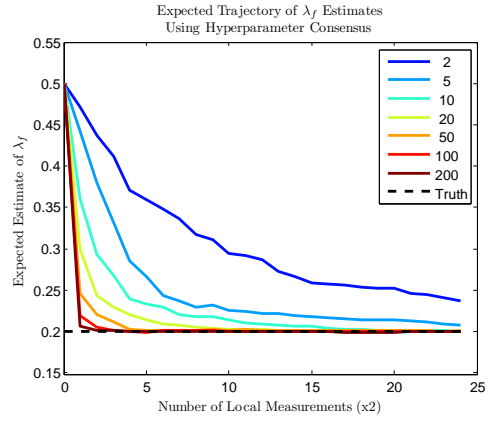


(f) Hyperparameter consensus on  $\mathbf{p}_n$

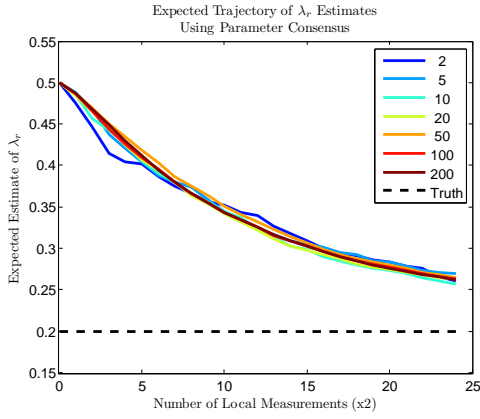
Figure 4-6: Expected probability estimates using each consensus method for varying number of local measurements



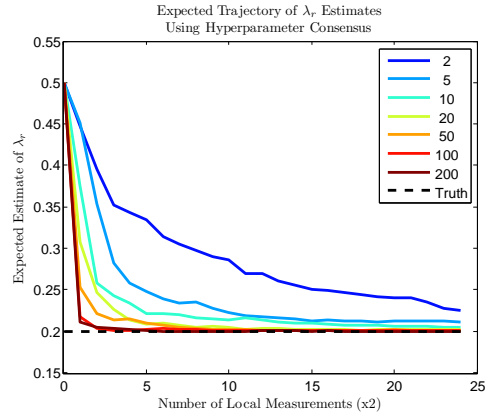
(a) Average consensus on  $\lambda_f$



(b) Hyperparameter consensus on  $\lambda_f$

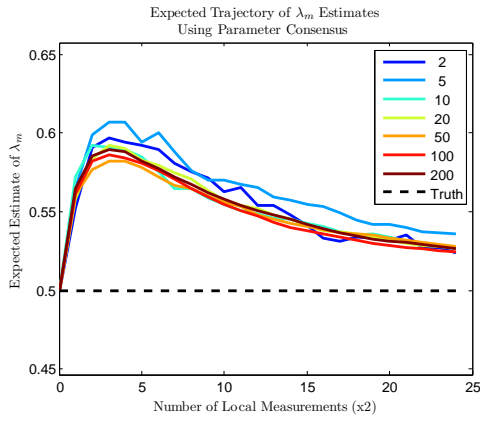


(c) Average consensus on  $\lambda_r$

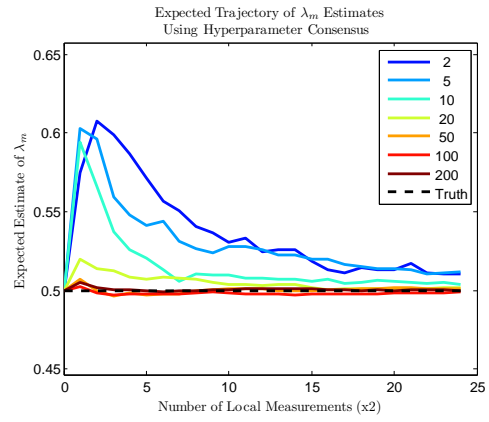


(d) Hyperparameter consensus on  $\lambda_r$

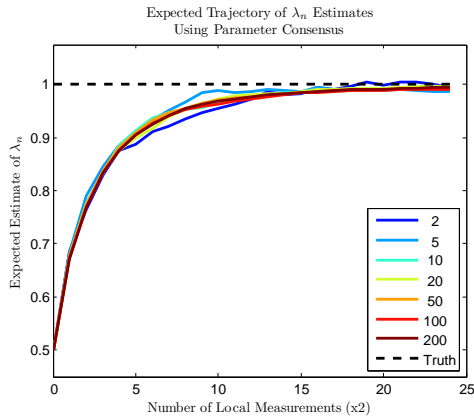
Figure 4-7: Expected  $\lambda$  estimates for each consensus method for varying number of local measurements



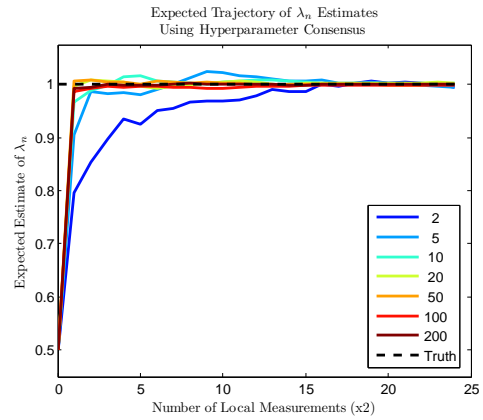
(a) Average consensus on  $\lambda_m$



(b) Hyperparameter consensus on  $\lambda_m$



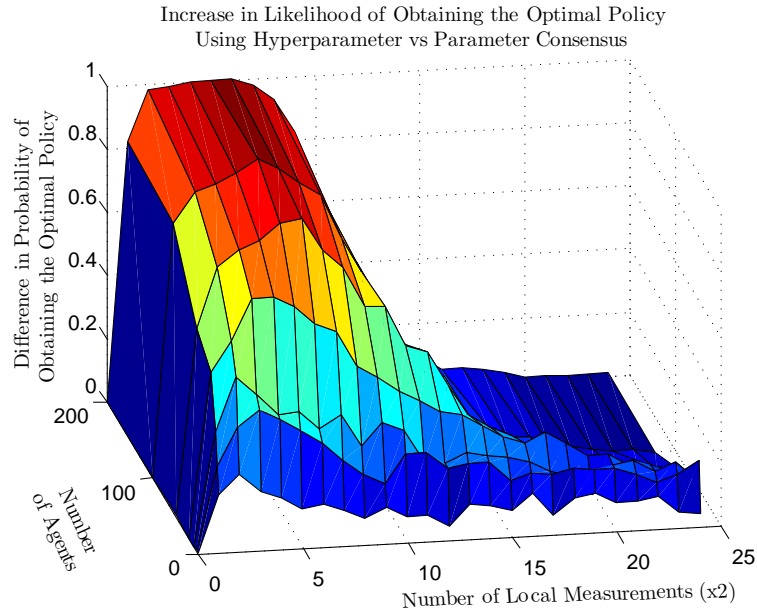
(c) Average consensus on  $\lambda_n$



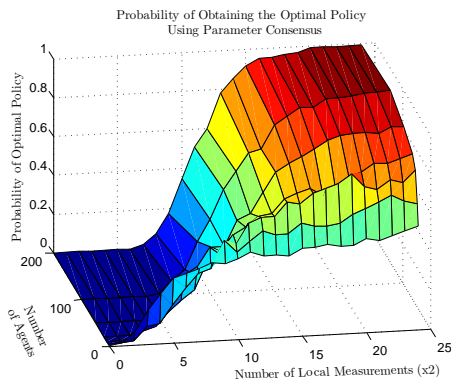
(d) Hyperparameter consensus on  $\lambda_n$

Figure 4-8: Expected  $\lambda$  estimates for each consensus method for varying number of local measurements

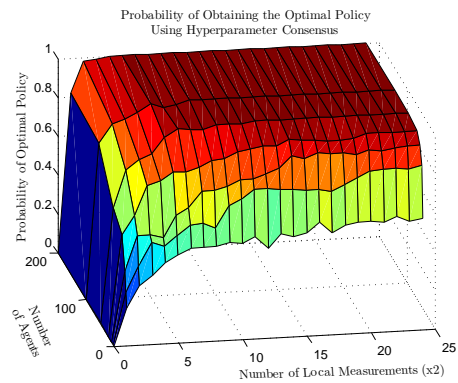




(a) Increased probability of obtaining the optimal policy using hyperparameter consensus over average consensus on the probabilities

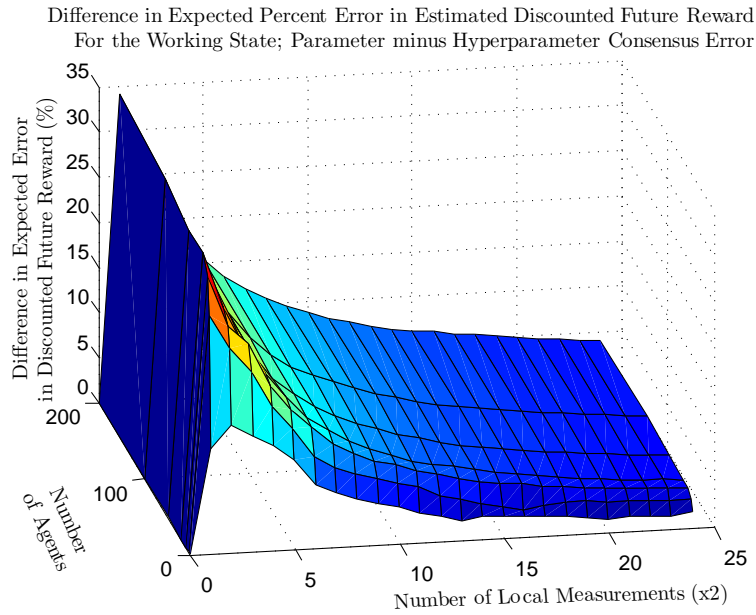


(b) Average consensus

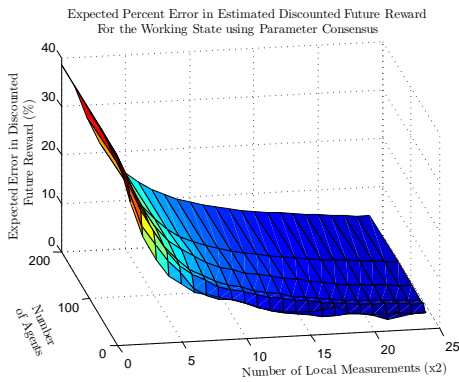


(c) Hyperparameter consensus

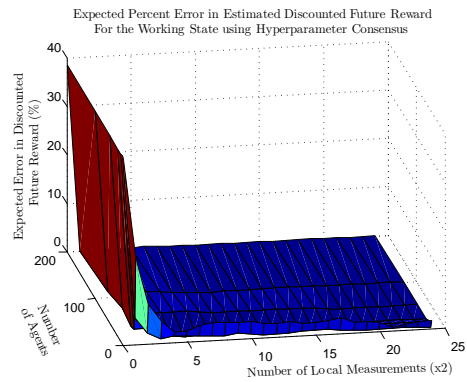
Figure 4-9: Probability of obtaining the optimal policy using each consensus method for varying number of local measurements



(a) Difference in expected error in the discounted future reward for the working state using hyperparameter consensus over average consensus on the probabilities

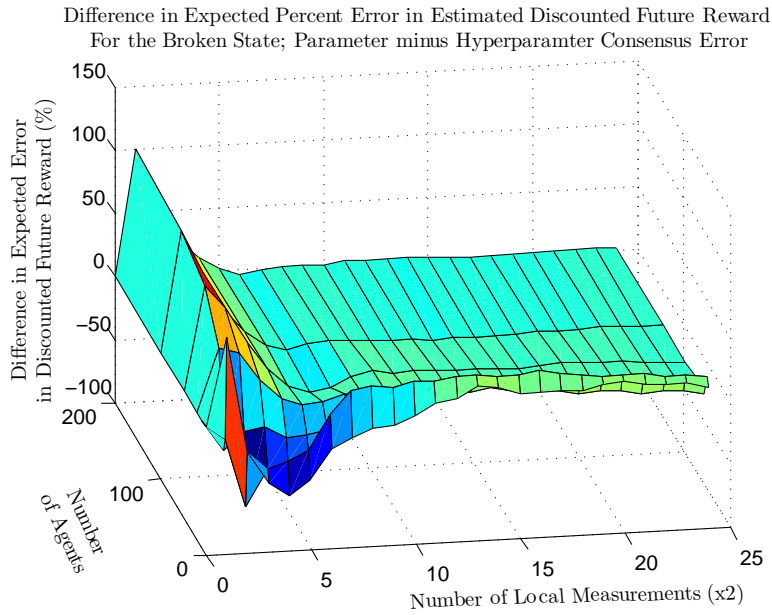


(b) Average consensus

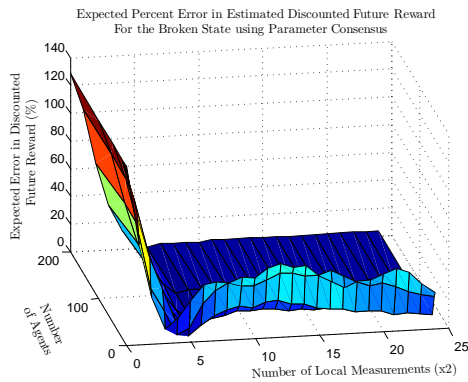


(c) Hyperparameter consensus

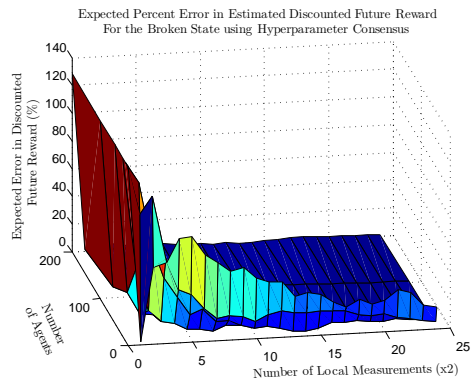
Figure 4-10: Expected error in the discounted future reward for the working state using each consensus method for varying number of local measurements



(a) Difference in expected error in the discounted future reward for the broken state using hyperparameter consensus over average consensus on the probabilities

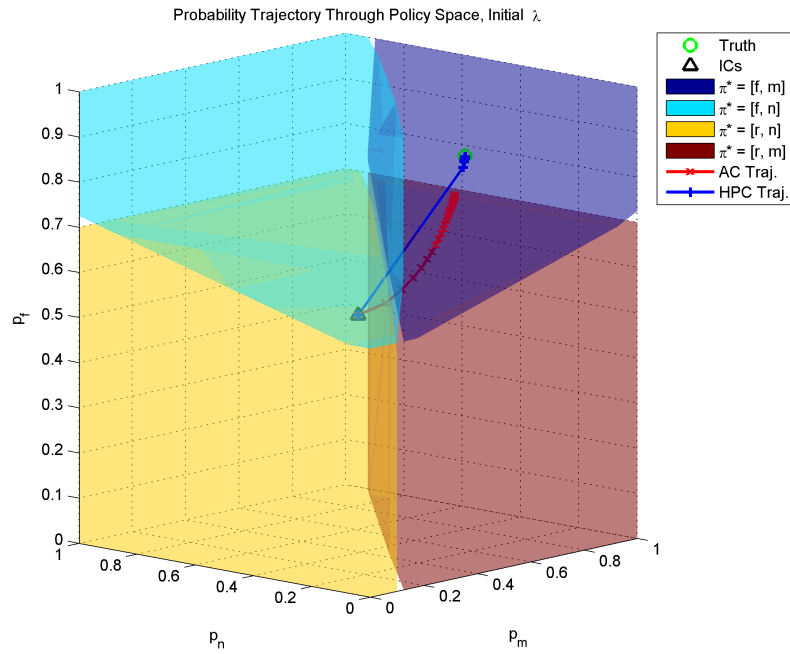


(b) Average Consensus

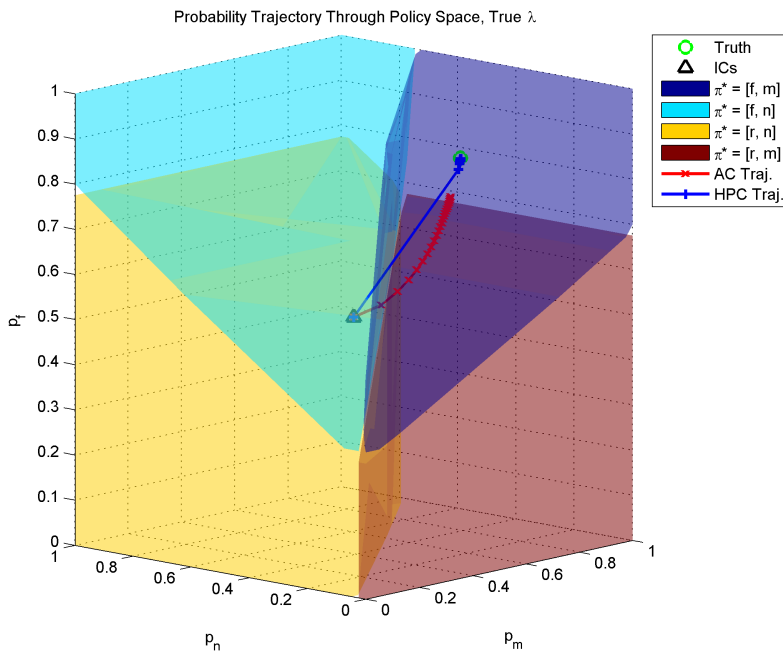


(c) Hyperparameter Consensus

Figure 4-11: Expected error in the discounted future reward for the broken state using each consensus method for varying number of local measurements



(a) Initial  $\lambda_a$  values



(b) True  $\lambda_a$  values

Figure 4-12: Trajectories of the probability estimates through probability space, superimposed over policy basins, for  $N = 200$ . Top: policy space for the initial  $\lambda$  estimates, bottom: policy space for the true  $\lambda$  values.

### 4.3 Summary and Future Work

This chapter has demonstrated the application of hyperparameter consensus to the problem of distributed estimation in the context of learning the underlying model of a multi-machine repair problem. In the proposed formulation, each agent locally learns the model through measurements that are taken independently of the other agents, and after a number of local measurements the agents are permitted to communicate to try and agree on the model. In order for the model estimation problem to not be biased towards learning only the actions dictated by the currently deemed optimal policy, each agent was set to execute a heavily exploratory policy, whereby the probability of selecting any given action from a state was roughly equal. It was shown that the hyperparameter consensus algorithm allowed the agents to effectively aggregate their *observations*, not just their current estimates, which led to greatly increased learning rates in larger networks. Conversely, average consensus on the current parameter estimates showed little to no improvement over the estimation power of a single agent, such that the pre- and post-consensus parameter estimates were roughly equivalent and generally much further from the truth than the equivalent post-hyperparameter consensus estimates.

Thus, the increased estimation power provided by the hyperparameter consensus method allowed the agents to converge to the true model with very few local measurements (though many agents), which, in turn, meant that the agents were more likely to obtain the true optimal policy and a good approximation of the expected discounted future rewards. In particular, the hyperparameter approach is able to limit the performance loss during the typical learning ‘transient’ period in which a single agent would be learning the model but has not yet converged to the true value. This also suggests that, in the context of the exploration vs exploitation trade-off, agents sharing information using the hyperparameter consensus method can make the transition to exploit the optimal policy much sooner than those sharing by averaging their models. This is possible by taking full advantage of all the measurements available to the combined network of agents through the aggregation of the hyperparameters.

Since, given enough time, each agent would have learned the model correctly independently of any communication, the benefits of hyperparameter consensus over average consensus (or no consensus) diminish after this learning transient period since it is assumed that all agents have accurately learned the model anyway. Finally, though the presented MDP was relatively simple, with two states and two actions per state, the methodology that was presented was independent of the size of the problem and so the presented trends can scale to much larger state spaces.

The results shown here assumed shared and, in some sense, “flat” prior distributions over the model parameters. This is not always the case, but served to place all the agents and simulations on the same initial footing. In practice, not all agents will have the same initial conditions, and determining what, if any, of the initial information is shared can be a difficult challenge. Typically, each node may start with a prior based off of assumptions and past experience, but these must be handled carefully. If an agent’s prior is biased heavily away from the true values, it could take a long time to correct for that bias, and may need the application of a fading factor to exponentially forget old information before being able to converge to the true value. The same holds true for the consensus case, where if multiple agents have unique unrepresentative priors, then the resulting hyperparameter consensus estimate, though having converged to the centralized estimate, will likely be biased from the true value that the agents are trying to estimate. Further, if multiple agents have the same representative prior (ie. a prior that has been determined by possibly observing a handful of outcomes, such that it is based on the true model but has not converged to the truth yet), but do not account for the fact that the prior is shared information, the resulting hyperparameter consensus estimate will be very confident in the prior information since it will treat each agent’s knowledge as independent and unique. Thus, in order for the hyperparameter consensus to increase the performance of an estimation problem, it is critically important that the information upon which the consensus is being run be treated properly.

An added benefit of using hyperparameter consensus that has not been fully addressed so far is that not only does each agent converge to the centralized parameter

estimate, but also to the centralized variance in the estimate. This fact means that all agents in the problem addressed in this chapter would not only have a consistent estimate of the model parameters, but also a consistent estimate of the uncertainty in the parameters. Dearden *et al.* [50] and Bertuccelli [7] have utilized the uncertainties in the transition models of MDPs to improve upon existing methods for the exploration vs exploitation trade-off and solution robustness, respectively. Since the hyperparameter consensus method preserves the uncertainty in the transition probability estimates, the approaches explored by these authors may now be applicable to cooperative multi-agent scenarios, especially in larger problems where the agents are not able to learn the environment as quickly as demonstrated in this chapter. Thus, agents may be able to take advantage of the distributed estimation afforded by hyperparameter consensus as well as improved exploration and robustness strategies to improve performance in larger-scale MDPs.





# Chapter 5

## CSAT in RAVEN

Increased UAV autonomy, particularly in the realm of autonomous search and track missions, remains at the forefront of much of the current research due to its intrinsic challenges. This chapter will highlight results from the hardware implementation of a Coordinated Search, Acquisition, and Track (CSAT) control architecture developed by Aurora Flight Sciences in conjunction with MIT in the Aerospace Control Lab's Real-time indoor Autonomous Vehicle test ENvironment (RAVEN)[59, 60] as presented by How *et. al.* in [61]. At the core of this approach is a decentralized planning algorithm that allocates the UAVs to different tasks in the environment. A key result of the solution is a synergistic combination of the search and track missions that enable the UAVs to periodically switch between modes of operation in order to both reduce the uncertainty in the unexplored portions of the environment, including finding lost or unknown targets, as well as to improve the certainty of tracked target estimates.

Much of the research presented thus far in this thesis was motivated by the results obtained from this testing. Primarily, the results prompted investigation on the twin concepts of adaptive tracking and search map agreement, which are discussed at the end of this chapter. These necessitated the initial investigations into agreement on uncertain parameters, such as estimated covariance matrices of normal distributions and large-scale probability vectors for the tracking and searching problems, respectively. Thus, following from the results of the hardware implementation, this chapter

will introduce some preliminary investigations into the adaptive tracking and search map fusion techniques to further extend the capabilities of the CSAT algorithm.

The chapter will begin with an overview of the CSAT architecture and the RAVEN testbed, followed by the implementation methodology used to mimic the distributed nature of the algorithm within the indoor environment. Some important results will be displayed from the hardware experiments that motivate the subsequent investigations of an adaptive tracking algorithm and search map consensus, with the goals of determining an optimal time to revisit tracked targets within the proposed CSAT framework while allowing for distributed tracking in communication-deficient environments. The chapter will conclude with simulated results of the proposed methods and suggestions for future work.

## 5.1 CSAT and RAVEN

The Cooperative Search, Acquisition, and Track (CSAT) mission requires an allocation of UAV assets to the potentially conflicting objectives of searching and tracking. While the searching component encourages exploration of the environment to maximize the probability of finding unknown targets, the tracking objective requires a vehicle to persistently focus on a single target. A successful mission will necessarily trade off between these two modes because it is generally undesirable to be in only a search or track mode throughout the course of the mission. Striking the right balance between these two objectives is of key importance for overall mission effectiveness since search must be performed throughout the course of the entire mission. One of the key results of the tested algorithm is that the dynamic transition between searching and tracking arises naturally from the problem specification, rather than being a behavior that is artificially encoded in the problem statement.

While many researchers (e.g. [62–64]) have examined algorithms for distributed task assignment problems, few such as Ref [65] have included strong experimental results demonstrating the implementation of these algorithms in tightly coupled missions such as search and track applications. Much of the previous work has primarily

emphasized one aspect of the mission (such as search [66, 67] or track [30, 37]), but little work has addressed a more synergistic combination of the two operating modes. A Bayesian framework for search and track was developed in [68], but did not consider the multi-vehicle task allocation problem. Elston and Frew [69] developed a hierarchical approach to the coordinated search and track mission, but based their revisit times on mean target motion, rather than explicitly propagating the uncertainty in the target's state.

### 5.1.1 CSAT

To achieve the goal of a decentralized search and track mission, the CSAT algorithm has been divided into a number of modules that communicate with each other over a network. Figure 5-1 shows the architecture for several vehicles and several targets. Each vehicle runs three modules onboard: the onboard vision module (OVM), the onboard planning module (OPM), and the autopilot module (APM). These work together to perform the sensing, planning, and control of each vehicle. The OVM takes in images from the vehicle's camera and provides a state estimate to the OPM for each target that it detects in the image. The OPM then either updates its own target state estimate based on the new information or propagates the estimate and uncertainty of the target if no measurements are available. If the vehicle does not find the target after a planned revisit, then the OPM reverts back to searching for the target, but aided by the knowledge of the target's last known position and velocity. The OPM uses these estimates and creates a plan for the vehicle to search for or track targets, as appropriate. The waypoints generated by the OPM are sent to the APM, which implements the plan by interfacing to the low-level controller on the flight vehicle. The APM also manages the vehicle's state estimate and distributes it to the other modules as necessary. The Target Manager (TM) generates commands for the targets and gathers their actual state information (rather than the estimate provided by the OVM), which are used as truth data for display in the User Interface (UI) and for subsequent analysis. The UI receives data from the OPM, APM, and TM and displays it in a bird's-eye view of the operations area.

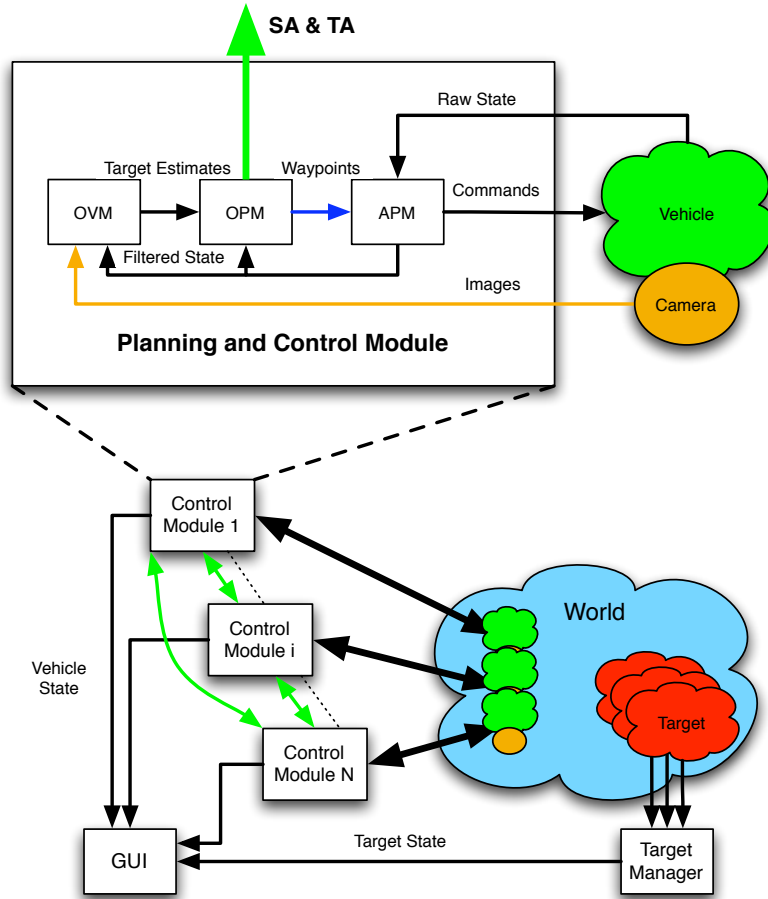


Figure 5-1: CSAT architecture block diagram

For each new vehicle, an additional instance of the OVM, OPM, and APM is executed. Since the vehicles are operating in a distributed manner, each UAV will only have direct access to its local information, which can lead to conflicts in the task assignment. To mitigate the effects of conflicting information, the OPMs on the vehicles communicate with each other to achieve global consensus on their plans and to coordinate their search and track efforts. This setup not only allows each vehicle to run its own algorithms, but allows separation of functions within the vehicle itself. For example, the high-level planner is a separate module from the low-level autopilot. The various modules communicate with each other over a TCP/IP network, which allows the modules to run on separate computers or simply as separate processes on the same computer. This modular approach adds additional robustness to the

system, allowing for the overall system to continue to execute the mission even if one or many modules fail.

## Onboard Planning Module

The onboard planning module (OPM) is the vehicle's high-level path and task planner. It assigns the vehicle to search for unknown targets or to track known targets. Whenever a vehicle is searching, the OPM uses a set of probability distributions to guide the vehicle along an optimal search path that maximizes the likelihood of detecting a target [66, 70–73]. Once a target is found and classified, it must then be periodically revisited to maintain an up-to-date estimate of its position and velocity [74, 75]. Between revisits, the OPM determines whether other tasks need to be executed or if the vehicle should resume searching the local area. This decision is made by a decentralized task assignment algorithm [76, 77] that continuously runs within the OPM on each vehicle and ensures that as many tasks as possible are executed without conflicting assignments. The following will describe first the search and track behaviors followed by how the two are allocated by the tasking algorithm within the OPM.

**Search:** To be able to search effectively [66, 70–73], the OPM maintains information about what areas of the mission environment have previously been observed. It does this by maintaining a set of probability maps where each cell  $(x, y)$  in the map  $M_i$  has a probability of containing a target  $P_t^i(x, y)$ , at some time  $t$ .<sup>1</sup> A generic probability map is maintained for each potential target environment encountered, such as land or water environments, and is initialized to represent any *a priori* knowledge of the unknown targets' position distributions. In addition, a new map is generated for each target that had been found at one point but has since been lost. This approach assumes that there may always be at least one additional undiscovered target in each environment type beyond those that have already been found. Thus, the generic search map provides the OPM with a constant incentive to continue searching the environment even if many targets have already been discovered and are successfully

---

<sup>1</sup>All  $P_t^i(x, y) = 0$  for any  $(x, y)$  in an obstacle

being tracked, ensuring that the UAVs maintain a nominal level of exploration of the environment.

When in search mode, the OPM uses the combined probability maps to determine a finite-horizon path that maximizes the sum of the probabilities that the sensor footprint will cover. This path generation scheme is based on a breadth-first tree search with limited depth and turn constraints, and includes not only the search path but the route to the next task, if applicable. If other UAVs are in the area, they will hierarchically coordinate their search paths so that they avoid searching the same area twice.

If a previously discovered target is lost, it is converted to a search target with an associated, newly created search map. This map is initialized with a non-zero probability only within an estimated reachable region based on the target's last known position and velocity, and is thereafter propagated based on the last estimate of the target's speed. To account for dynamic search targets that have not been found yet, each generic search map is associated with a phantom search target with its own velocity estimate to be used to determine how quickly uncertainty diffuses back into previously explored space. The probability diffusion update for each cell  $(x, y)$  in map  $M_i$  at time  $t$  is given by the two stage process

$$P_t^i(x, y) = \left( P_{t-1}^i(x, y) + P_{t-1}^i(x', y')P(x, y|x', y', v_i) \right) (1 - sc) \quad (5.1)$$

$$P_t^i(x, y) \leftarrow \frac{P_t^i(x, y)}{\sum_{\hat{x}, \hat{y} \in M_i} P_t^i(\hat{x}, \hat{y})} \quad (5.2)$$

where  $P(x, y|x', y', v_i)$  is the probability that a target transitions from  $(x', y')$  to  $(x, y)$  given the search target velocity  $v_i$ , and  $sc$  is the percentage of the cell that is covered by the sensor at time  $t$ . Equation 5.1 is the diffusion process while Eq. 5.2 ensures the sum of the probabilities is unity. By diffusing the probabilities as such, the OPM promotes searching for dynamic targets that may have traveled back into previously explored territory.

**Track:** Once a target is successfully detected by the vision module, it is classified

and tracked by a UAV with the required capability. The OVM provides target state measurements to update the OPM’s internal estimator. In our experiments, a Kalman filter was used under the assumption that the dynamic system is linear [74]. Target tracks are maintained by recursively updating the state estimates  $\hat{X}_{k+1|k}$  using a kinematic model  $A$  of the target motion with additive noise  $w_k \sim \mathcal{N}(0, Q)$  to capture any unmodeled dynamics due to this simplification

$$\begin{aligned} \textbf{True Model: } X_{k+1} &= A X_k + B w_k, \\ \textbf{Estimate: } \hat{X}_{k+1|k} &= A \hat{X}_{k|k} \end{aligned} \tag{5.3}$$

Once the uncertainty in the target estimate has been reduced through tracking measurements, the UAV can temporarily leave the target to execute other tasks and be confident that the target can be re-acquired upon its return. This permits each vehicle to complete multiple tasks even if the number of available tasks exceeds the number of capable agents.

In order to determine the necessary revisit time, Algorithm 3 is run for each track-capable UAV (due to possibly differing sensor footprints). The target estimate and error covariance are both propagated forward until a scaled representation of the covariance ellipse no longer can be contained within the vehicle’s sensor footprint.<sup>2</sup> The scaling multiplier used, denoted  $n_\sigma$  in Algorithm 3, can be thought to represent a desired confidence level on finding the target at the revisit time and location, with higher values leading to more conservative revisit times. UAVs with different sensor footprints will in general predict a unique revisit time based on their physical sensor properties. A track task is then created to visit the target at the revisit time and the target’s propagated position using a vehicle capable of tracking.

When a vehicle is assigned to track a target, the OPM generates a path that coordinates the vehicle arrival time at the revisit location to match the predicted revisit time. Upon UAV arrival at the desired location, the target may or may not

---

<sup>2</sup>This forward propagation uses the existing process noise to propagate the error covariance by recursively using the prediction step of the Kalman Filter and is measurement independent.

---

**Algorithm 3** Revisit time and location calculation

---

```
 $k \leftarrow 0$   
Initialize process noise covariance  $Q$   
Initialize state estimate  $\hat{X}_{k|k} \leftarrow \hat{X}_0$   
Initialize error covariance  $P_k \leftarrow P_0$   
Initialize characteristic size  $\phi_k \leftarrow \pi\sqrt{|P_k|}$   
while  $\phi_k < \phi_{sensor}$  do  
   $\hat{X}_{k+1|k} \leftarrow A \hat{X}_{k|k}$   
   $P_{k+1|k} \leftarrow AP_{k|k}A^T + BQB^T$   
   $\phi_{k+1} \leftarrow \pi\sqrt{|P_{k+1|k}|}$   
   $k \leftarrow k + 1$   
end while  
return Revisit time =  $k$ 
```

---

be within the UAV’s field of view. In the first case, the vehicle overflies the intended target for a predetermined time, keeping the target in its field of view, and updates its position and velocity estimates. The specified track time is an empirically chosen value that was determined to be long enough to obtain a reliable state estimate of the target, though both the tracking trajectory and duration can be modified to incorporate any desired tracking algorithm. In the second case, however, the target is declared “lost”, and a new search is initialized. If the target is once again found, the search probability map is removed and a new revisit location and time is calculated.

### Task Assignment Algorithm

Given the updated search probabilities maps and target estimates, the OPM can then decide whether to search regions of the map that have a high likelihood of containing targets, or execute existing track tasks. In our framework, search is considered a “spare time strategy” rather than a task. This means that the vehicles search for targets when they are not assigned to a track task or when their next track task is far enough in the future that they can search in the intervening time. This approach assumes that, given the choice between keeping track of a known, non-trivial target and searching, it is more beneficial to follow the targets that have already been found.

The OPM uses a modification of a multi-agent task assignment algorithm introduced in Refs. [76, 77], called the Consensus-Based Bundle Algorithm (CBBA).



CBBA is a cooperative, low-communications-bandwidth iterative auction approach that uses two phases to achieve a conflict-free task assignment. In the first phase, each vehicle generates a single ordered “bundle” of tasks by greedily selecting tasks for itself. The second phase resolves inconsistent or conflicting assignments through heuristic methods, and improves the global reward through the bidding process. The implementation of CBBA in the OPM executes these two phases continuously and concurrently at each time step, allowing the algorithm to rapidly adjust to changes in the network and environment. See [76, 77] for additional details.

### **Onboard Vision Module**

Images captured from the onboard camera are loaded and analyzed by the vision processing unit using OpenCV [78]. The received image is converted from an RGB (Red-Green-Blue) format to a HLS (Hue- Lightness-Saturation) format for easier color separation. Then, given the expected color ranges for each target, a detection algorithm determines which pixels fall within the range of colors for each target, and a smoothing function is applied to each color ‘blob’ to locate its centroid. The location of the target in the image plane is then projected to the inertial world frame using a calibrated pinhole camera model, assuming that targets exist on the ground plane ( $z = 0$ ). This estimate of the targets’ locations is then fed into a particle filter to smooth the measurement before transmission to the OPM.

Target state estimation relies on principles from particle filtering [79]. Upon receiving a (noisy) measurement of the location of the target in the inertial world frame, each particle’s location is updated using the kinematic motion model of the target. The particles are then re-weighted based on their distance from the target location as measured from the camera image, and importance re-sampling is performed on the set of particles [79]. Particles with low weight are rejected, and new particles are generated. At this point, the set of particles should approximate the distribution of possible target states, and the weighted mean value is transmitted to the OPM as the new target measurement.

## Autopilot Module

The autopilot module (APM) acts as the interface between the low-level vehicle controller and the rest of the CSAT architecture. In simulation modes, it also simulates vehicle dynamics. Specifically, the autopilot maintains the vehicle state estimate, provides guidance to fly the vehicle along the waypoints provided by the OPM, and monitors the health of the vehicle, including fuel status.

The APM's open architecture allows it to accept state estimate input from various sources depending on the situation. For example, in simulation it simply takes the state estimate from the simulated dynamics, while in flight experiments it might use state estimates from on-board or off-board sensors. The APM can also perform additional filtering on the state information. This state estimate is then distributed to the other modules that need the estimate, including the OPM and the OVM, for use in planning and target estimation.

The APM periodically receives a list of waypoints from the OPM that describe the planned path over a short time horizon. If the APM is using simulated dynamics, it generates appropriate steering commands using the nonlinear control law developed in [80]. If a separate vehicle controller with waypoint following ability is used, such as in the RAVEN testbed, then the APM only keeps track of which waypoint the vehicle should fly to next using logic developed in [81] and sends that waypoint to the vehicle controller. Waypoints are specified as a position, a time at which the vehicle should reach that location, and a type, such as fly-by, fly-over, or stop, that specifies when the vehicle can continue on to the next waypoint.

### 5.1.2 RAVEN

Experimental trials of the CSAT algorithm were conducted in MIT's RAVEN (Real-time indoor Autonomous Vehicle test ENvironment) [59, 60], a multi-vehicle platform allowing for rapid-prototyping of high-level mission management algorithms. This capability is achieved by using one of two very accurate motion capture systems (the Vicon MX system [82] and the Motion Analysis Raptor 4 digital real-time system [83])

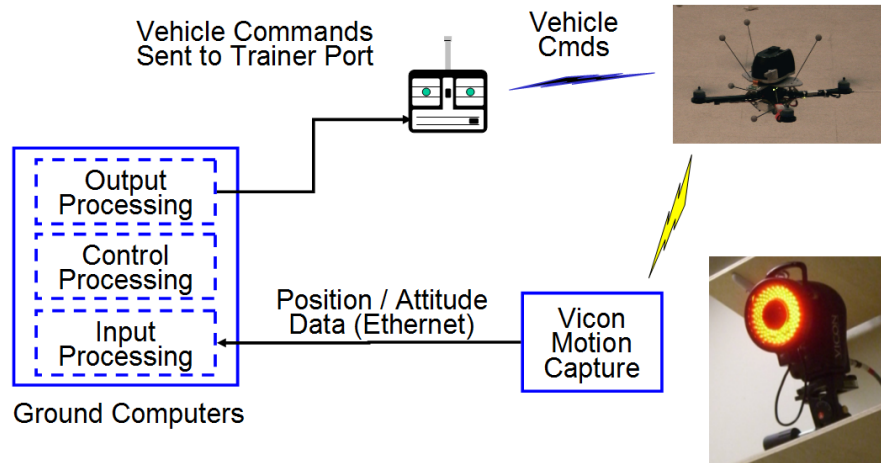


Figure 5-2: RAVEN architecture

to produce high bandwidth state estimates of numerous aerial and ground vehicles, as well as in-house vehicle controllers to provide low-level control and stabilization of the vehicle hardware.

The system architecture is displayed in Figure 5-2. The motion capture system detects lightweight reflective dots on the vehicles (as can be seen on one of the quadrotor aircraft in the top right of the image) and uses these to calculate the vehicles' position and orientation within the 25 by 30 foot test room. This data is transmitted via ethernet to each vehicle's ground based control computer, which in turn commands its vehicle through a commercial, off-the-shelf (COTS) radio control (R/C) transmitter [59, 60]. The primary reason for this configuration is that the bulk of the sensing and control computation is moved off-board into the computational infrastructure, thus avoiding the risks associated with using expensive on-board sensors and equipment and allowing for commercially available vehicles with and, especially, without complex onboard computational capabilities to be quickly integrated into the system.

While the RAVEN testbed allows for a wide range of vehicle types to be used, the room space constraints and prior proven vehicle performance led to the selection of the Hummingbird quad rotor produced by Ascending Technologies [84] as the aerial vehicles (see Figure 5-3). The particular model used can stabilize the vehicle

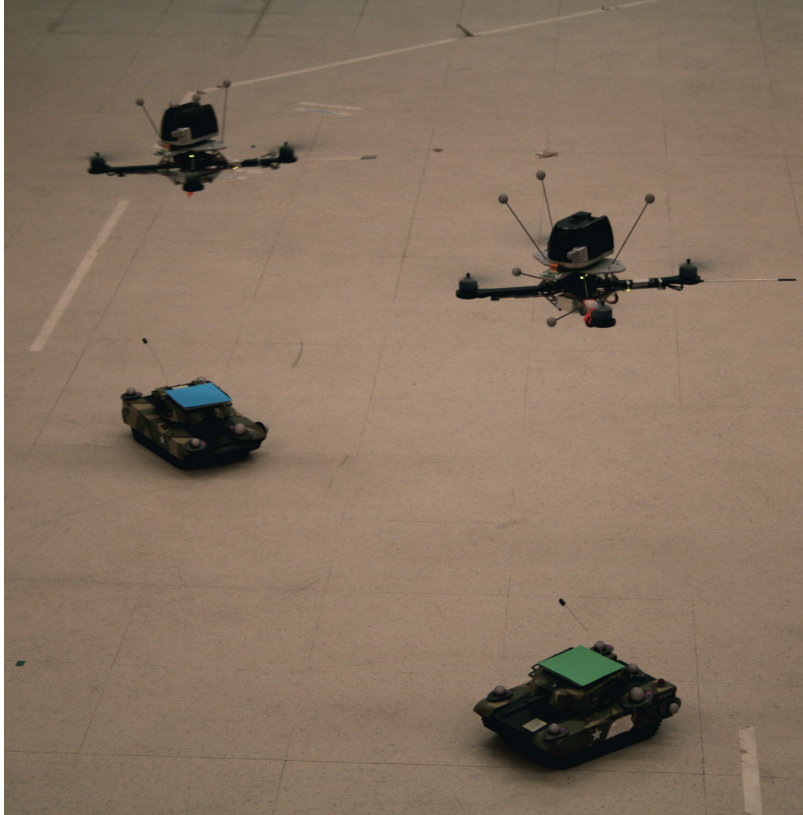


Figure 5-3: Modified Hummingbird UAVs performing a coordinated search and track task on tank targets

attitude using onboard sensors and microcontrollers while an associated mid-level control program running on one of the RAVEN vehicle computers generates attitude commands to control the position of the vehicle. This *vehicle wrapper* code is the link between the Vicon motion capture state estimates, the CSAT Autopilot module, and the vehicle itself. The wrapper implements a simple LQR controller to follow a reference trajectory generated by an internal waypoint follower [60]. The APM sends Activate, Take-off, Land, and Waypoint commands to the wrapper, which are then converted to the relevant control signals to send to the vehicle.

The Hummingbirds are modified with the COTS Wi-Fi enabled Panasonic BL-C131A network camera to provide the vehicles with visual detection capabilities. The camera is networked with the OVM module of the CSAT framework. The internal mechanisms to control the pan and the tilt of this camera are removed in order to reduce the overall payload weight, making the current hardware configuration

constrain the cameras to a fixed orientation looking vertically downward from the vehicle. This is a 1/6" CMOS sensor, with approximately 320,000 pixels that provides a footprint of 1.0[m] wide by 0.75[m] high at an altitude of 1.2[m].

## 5.2 Implementation

The primary focus during the integration of the CSAT algorithm with the RAVEN system was on maintaining the distributed nature of the algorithm within the confined space. In a full-scale application the algorithms would be running onboard each vehicle in the fleet; however, the small COTS aircraft that are used in the RAVEN environment lack the computational ability to run the algorithms onboard. Since a key feature of the algorithm is its scalability and decentralized assignment algorithm, it was necessary to maintain some measure of this distributed nature. This was achieved by assigning each vehicle to a dedicated control computer within the system that would act in place of an onboard processor. Therefore, along the same lines as the low-level controllers, the typically onboard modules (the OPM, APM and OVM) would run off-board but on a dedicated computer. This off-board computation replicates the exact type of computation that would be performed onboard each vehicle, and it is performed off-board simply to ease the integration process given the payload restrictions of the current vehicles while preserving the distributed nature of the algorithm.

For hardware implementation, a few modifications from the proposed architecture were missing. First, the Autopilot module needed to be modified to access state information from the Vicon data stream and send the appropriate waypoint messages to the low-level vehicle wrapper. Second, the OVM was modified to poll the network camera for images and process them to track colored identifiers placed on each target.

## 5.3 Results

This section presents some the results of the CSAT architecture flown in RAVEN. The indoor flight experiments used 5 targets and 3 UAVs, a limitation imposed only by the physical size of the indoor test environment. In order to scale the speed of the experiment to the size of the environment, the UAVs were flown with a maximum speed of 0.2 m/s at an altitude of 1.2 m. Some of the targets were stationary, while the dynamic targets were controlled autonomously and by R/C and designed for a nominal speed of 0.05 m/s.

The following scenario demonstrated a multi-vehicle, multi-target mission with 3 autonomous UAVs, and five targets (two of which were dynamic). In order to model uncertainty in the targets, their estimated process noise covariance matrices were set to  $Q = \text{diag}(0.001)[m^2/s^4]$  for any target with measured velocity less than 0.005m/s, and  $Q = \text{diag}(0.05)[m^2/s^4]$  for those with velocities over 0.005m/s. Also, Target 2 was given a higher tracking score and desired confidence level ( $n_\sigma > 1$ ), specifying that it is a high priority target that the agents need to track if found. The CSAT planner's search map was initialized with a uniform prior distribution of target locations. All vehicles in this experiment were capable of both searching for and tracking targets.

Figure 5-4 shows a summary of the mission using four metrics: (clockwise from top left) the vehicle trajectories, vehicle states, area searched and targets tracked. In general, this mission shows a good balance between searching and tracking, as well as alternating between tracking erratic dynamic targets and static targets. The top left figure shows the trajectory of the three UAVs (the apparent noise in the paths is due to perturbations from the downwash effects of the multiple UAVs rather than the algorithm). We can see from the overlapping trajectories that the UAVs are using a fluid search method rather than partitioning the operations area or flying fixed "zamboni" patterns. The advantage of this approach is that the team is inherently flexible and the agents can explore regions of high uncertainty regardless of their location, as opposed to remaining constrained to local areas or inefficient search paths. Additionally, as agents are sent to complete other tasks, nearby searching vehicles can

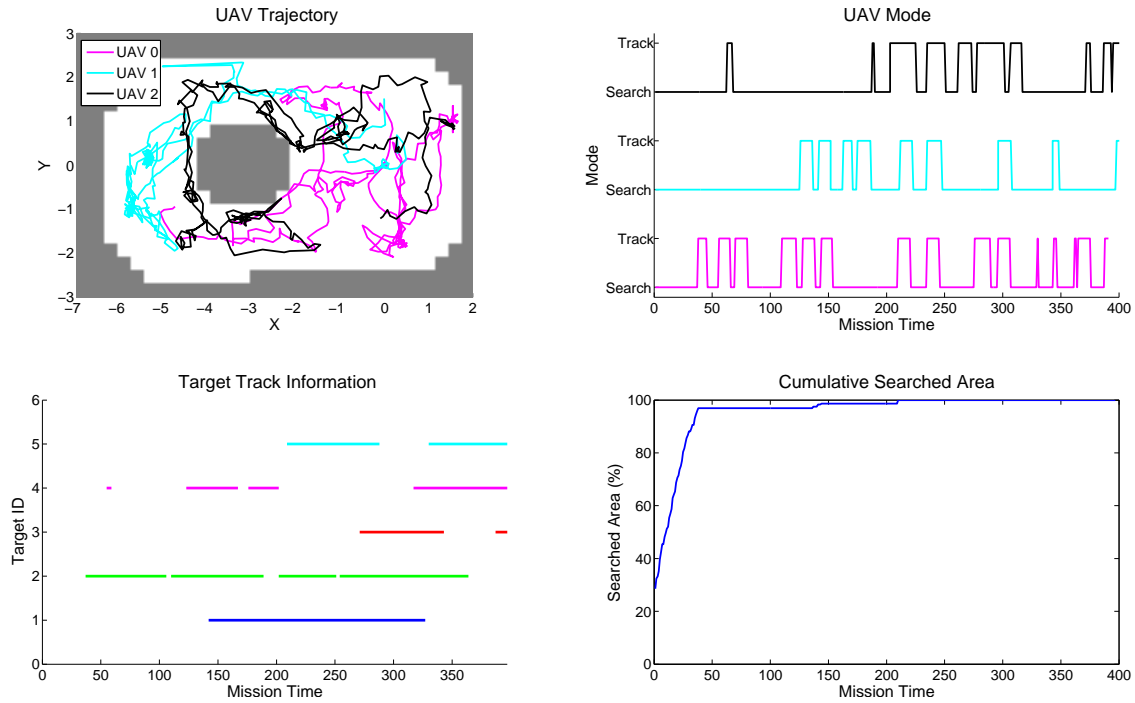


Figure 5-4: CSAT mission performance. Clockwise from top left: Overhead view of UAV trajectories, UAV modes (search or track), total percentage of the environment searched, and times at which each target estimate was accurate (line means accurate estimate)

take over in the newly vacated region.

The top right plot in Figure 5-4 shows the mode (search or track) of each UAV. Since the environment has a mix of dynamic targets (with high covariance) and static targets (with low covariance), the UAVs exhibit both short and long revisit times. This permits agents to naturally execute target hand-offs if one agent needs to service another target or refuel. This plot also demonstrates the natural shift of focus from primarily searching early on to primarily tracking as more targets are found.

The bottom plots in Figure 5-4 demonstrate this shift even more clearly. In the bottom right graph, it can be seen that the UAVs have enough time to search, despite the track tasks, close to 100% of the map. The bottom left graph shows when the estimate of each target is within a specified threshold of the target's true position. All five targets were found, tracked, and revisited during the mission. As was intended, UAV 2 (in green) is tracked more frequently due to its higher priority and higher

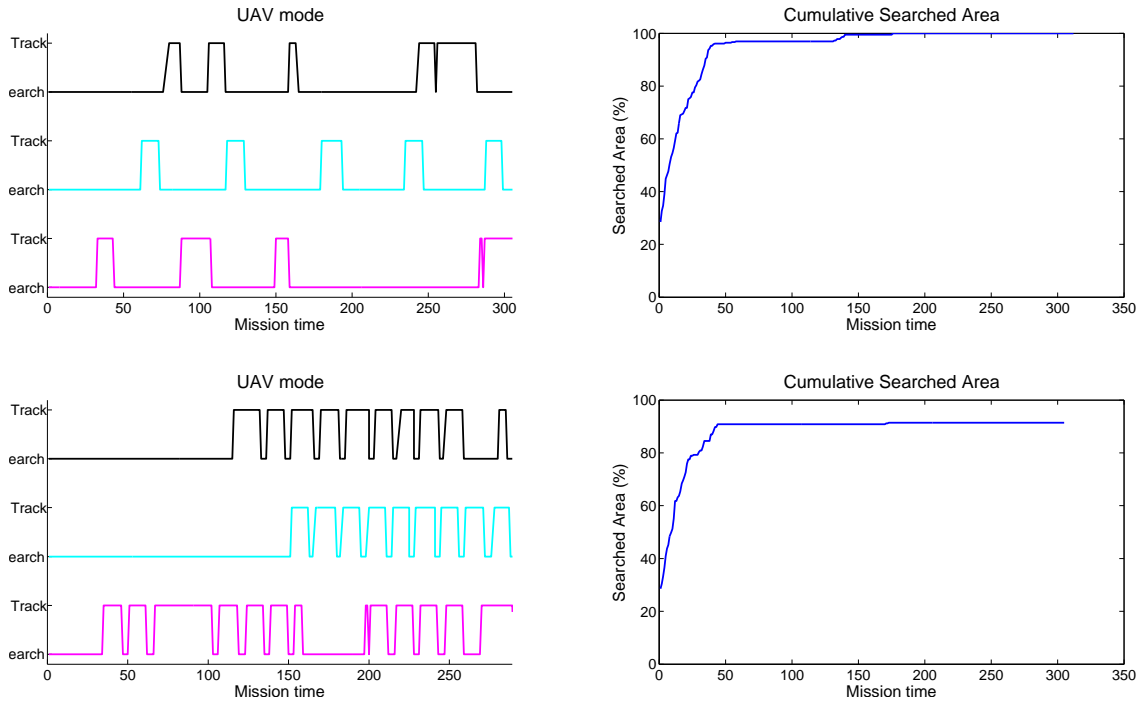


Figure 5-5: Comparison of (top) low  $Q$  value and (bottom) high  $Q$  value results: UAV mode (left) and percentage of area searched (right). Low  $Q$  promotes long revisit times and allows for searching for new targets, while high  $Q$  encourages more continuous tracking of known targets

desired revisit confidence. This results in the UAVs maintaining a better estimate of that target than of the other targets, though the agents were all able to contribute to the search and track components for the duration of the mission.

### Need for Tuning the Process Noise Covariance

Two additional tests were conducted using 3 UAVs and 5 targets in which the goal was to investigate the sensitivity of the CSAT planner to the choice of the process noise covariance  $Q$ . Under the assumption of a target motion model driven by process noise, the noise covariance is used to determine the revisit time for the tracking exercise as per Algorithm 3. The process noise covariance is an effective tuning parameter for the CSAT controller that must be chosen carefully, based on anticipated target maneuverability. Thus, the true  $Q$  will inherently vary between vehicles, specifically between static and mobile targets, according to the vehicle dynamics.



Two choices of  $Q$  were made in this scenario resulting in two distinct experiments (the scaling factor  $n_\sigma = 1$  was used for both experiments). The results are shown in Figures 5-5. The top plots show an experiment with low process noise covariance ( $Q = 0.001[m^2/s^4]$ ) for all targets, while the bottom plots show experiments with high covariance ( $Q = 0.05[m^2/s^4]$ ) for all targets. In the first experiment, the low value of  $Q = 0.001[m^2/s^4]$  implied that the targets being tracked were assumed to not be very maneuverable. Because of this, the UAVs track the targets occasionally, but spend most of their time searching since they are confident they can re-locate the target upon revisit. Consequently, the cumulative area searched quickly approaches and eventually reaches 100%. Conversely, in the second experiment, where a high value of  $Q = 0.05[m^2/s^4]$  was used, the motion of the targets is assumed to be very uncertain and results in the targets being revisited often. Once a UAV finds a target, it spends most of its time in tracking mode and only has a few seconds of search between each revisit. This is reflected in the cumulative search area, which quickly plateaus once the first UAV begins tracking, and never reaches the same level as in the low  $Q$  experiment. Furthermore, because the UAVs in the high  $Q$  situation have so little time to search between revisits, they can never wander far from their assigned targets. As a result, they will have difficulty finding any new targets entering the operations area. Essentially, the area searched after tracking has begun is much lower in the second case than in the first.

Even in the restricted testing space, the experiments showed that a small change in the choice of the process noise  $Q$  can lead to significant variations in the tracking strategies. In particular, if the target dynamics are unknown but likely very erratic, as would be the case with highly maneuverable targets, it is probably a good choice to select a high  $Q$  and emphasize tracking at the expense of search performance. Conversely, if the choice of dynamic model is assumed very accurate or the expected targets are not very maneuverable, then a matching low  $Q$  value will allow the agents to execute other tasks while proceeding with the track. Thus, it is very beneficial to the overall execution of the CSAT mission to choose a representative  $Q$  for each target to best trade off between the two competing tasks. However, it is often difficult

to accurately estimate *a priori* the process noise expected in a system, especially if the system to be estimated is inherently unknown. This difficulty gave rise to the idea of an adaptive tracking algorithm that learns from the observed process noise, which will be discussed next.

### 5.3.1 Process Noise Adaptation

The key ability of the demonstrated CSAT algorithm that allows for good performance in both searching and tracking regimes is the concept of leaving known track targets to pursue other goals, with the assumption that the target can be re-located at a certain revisit time in the future. This periodic tracking ability is desirable in many situations, such as when the number of track targets is greater than the available resources or when it is beneficial that the sensing equipment be used sparingly due to energy requirements or increased visibility to hostile targets during tracking. In the situation considered here, each sensor must be able to allocate its time intelligently among the targets, and ideally be able to track multiple targets by switching or cycling through the tracks.

To accomplish this periodic tracking ability, the estimation of the target state must be of sufficient fidelity that the sensor can stop tracking it for a period of time and be able to resume the track at a later time, with enough time elapsed between the end of the track and the revisit to conduct other meaningful tasks. In other words, the problem is to maximize the “down time” of the sensors subject to the constraint that, at the revisit time and given some mapping between measurements and a state estimate, the true state will be within that sensor’s footprint, centered at the sensor’s estimate, with probability greater than  $\eta$ . This formulation desires that the revisit time is as late as possible, while still providing some bound on the probability of re-locating the target. This gives rise to the formulation:

Given measurements from time 1 to  $t_{track}$ ,  $Z_j^{t_{track}} = [z_j^1 z_j^2 \dots z_j^{t_{track}}]$ , where each  $z_j^t$  is the set of all measurements of target  $j$  at timestep  $t$  by all sensors that are tracking target  $j$  at that time, and given some estimation

procedure for sensor  $i$  that maps the measurements up to time  $\tau$  to an estimate,  $\hat{x}$ , at time  $t$ ,  $\mathbb{E}_{i,j}^{t|\tau} : Z_j^\tau \mapsto \hat{x}_{i,j}^t$ ,  $t \geq \tau$ , maximize the revisit time  $\Delta t_{revisit}^{i,j}$  for each agent such that the probability of re-locating the target at  $t_{revisit}^{i,j}$  is greater than some threshold,  $\eta$ :

$$\begin{aligned}
& \max \quad \sum_{i \in \text{sensors}} \sum_{j \in \text{targets}} \Delta t_{revisit}^{i,j} \\
& \text{s.t.} \quad Pr(|\tilde{x}_{i,j}^{t_{revisit}^{i,j}}| \leq r_{sensor}^i) \geq \eta \\
& \quad \tilde{x}_{i,j}^t = x_j^t - \hat{x}_{i,j}^t \\
& \quad \hat{x}_{i,j}^{t_{revisit}^{i,j}} = \mathbb{E}_{i,j}^{t_{revisit}^{i,j} | t_{track}}(Z_j^{t_{track}}) \\
& \quad t_{revisit}^{i,j} = t_{track} + \Delta t_{revisit}^{i,j}
\end{aligned}$$

Where, in the first condition, only the states that are physically measured by the sensor (and therefore must be within its footprint,  $r_{sensor}^i$ ) are considered, the magnitude operates element-wise, and  $Pr(A)$  denotes the probability of the event  $A$ .

For an arbitrary value of  $0 \leq \eta < 1$  and  $t_{track}$ , and a target that follows a linear-Gaussian process, the solution to this problem can be achieved using a true model of the system with accurate noise characteristics, an information fusion scheme between the sensors, and an intelligent way to determine the revisit time. Information fusion can be handled in the Kalman filter framework using hyperparameter consensus on the local  $Y$  and  $y$  values between state propagation stages or by utilizing methods similar to those in [34, 39, 43] (Kalman Consensus is inappropriate here due to the biased error covariance  $P$  as demonstrated in Section 2.3.3), though for the following discussion it will be assumed that there is only one agent tracking the target at a time to avoid this complexity. The revisit calculation in Algorithm 3 will be shown to give the appropriate revisit time, but only if the estimation parameters are known fully. Therefore, the outstanding problem to be addressed is knowledge of the model parameters. In this case, it is assumed that a good knowledge of the system transition model exists and that the main uncertainty is in the process noise. Thus, by adapting to the process noise, the following sections will show that the algorithm can reach an

approximate solution to the problem as defined above.

### Tracking a Target on a Line

To examine the proposed concept, investigations were conducted on the 1-D case. The assumed dynamics of the target amount to the typical double integrator:

$$\ddot{x} = u + w$$

which, in state space form, gives:

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w$$

or, in discrete time, becomes:

$$\begin{bmatrix} x_{t+1} \\ \dot{x}_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} + \begin{bmatrix} dt^2/2 \\ dt \end{bmatrix} u_t + \begin{bmatrix} dt^2/2 \\ dt \end{bmatrix} w_t$$

Assuming that the control input is unknown and can be modeled from the estimator's point of view as additional process noise, the estimated dynamic equations for the state are:

$$\begin{bmatrix} x_{t+1} \\ \dot{x}_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_t \\ \dot{x}_t \end{bmatrix} + \begin{bmatrix} dt^2/2 \\ dt \end{bmatrix} w_t \quad (5.4)$$

$$X_{t+1} = AX_t + Bw_t \quad (5.5)$$

$$z_t = H_t X_t + v_t \quad (5.6)$$

The typical Kalman filter prediction equations are:

$$\hat{X}_{t+1|t} = A\hat{X}_{t|t} \quad (5.7)$$

$$P_{t+1|t} = AP_{t|t}A^T + BQ_tB^T \quad (5.8)$$

and the update equations are:

$$\begin{aligned}
K_{t+1} &= P_{t+1|t} H_{t+1}^T (H_{t+1} P_{t+1|t} H_{t+1}^T + R_{t+1})^{-1} \\
\hat{X}_{t+1|t+1} &= \hat{X}_{t+1|t} + K_{t+1} (z_{t+1} - H_{t+1} \hat{X}_{t+1|t}) \\
P_{t+1|t+1} &= P_{t+1|t} - K_{t+1} H_{t+1} P_{t+1|t}
\end{aligned}$$

This research proposes the additional update step of adapting to the process noise covariance matrix,  $Q$ , through a recursive method as suggested by [85, 86]. The method updates the estimate of the process noise,  $\hat{Q}$ , after each sensor measurement as:

$$\hat{Q}_{t+1}^* = B^{-1} ((\hat{X}_{t+1|t+1} - \hat{X}_{t+1|t})(\hat{X}_{t+1|t+1} - \hat{X}_{t+1|t})^T + P_{t+1|t+1} - A P_{t|t} A^T) B^{-T} \quad (5.9)$$

$$\hat{Q}_{t+1} = \begin{cases} \hat{Q}_t + \frac{1}{L} (\hat{Q}_{t+1}^* - \hat{Q}_t) & \text{if the result is } \succ 0 \\ \hat{Q}_t & \text{otherwise} \end{cases} \quad (5.10)$$

where  $L$  is a user-selected gain. In the above equations, if  $B^{-1}$  doesn't exist, the pseudo-inverse can be used, and the  $\succ 0$  denotes positive-definiteness of a matrix. Thus, the  $\hat{Q}_{t+1}^*$  represents, in some sense, the MLE of the process noise covariance matrix given the observation at time  $t+1$ , and the positive definiteness of the resulting  $Q$  estimate is ensured through the second equation.

## Revisit Calculation

In the considered case where a revisit time and location must be made for a track task since continuous tracking is not desired or not possible, the determination of this revisit is highly sensitive to both the assumed process and measurement noise values ( $Q$  and  $R$ ), the size of the sensor footprint ( $r_{sensor}$ ), and the desired confidence level ( $n_\sigma$ ). The basic determination of the revisit location is to consider when the propagated state error covariance matrix (given by a recursion of Eq. 5.8) has diluted the confidence in the accuracy of the estimate such that the uncertainty ellipse associated with the estimate is of the same size as the sensor's footprint. In other

---

**Algorithm 4** 1-D revisit time and location calculation

---

```
 $k \leftarrow 0$   
Set process noise covariance  $Q \leftarrow \hat{Q}_{t_{track}}$   
Set state estimate  $\hat{X}_k \leftarrow \hat{X}_{t_{track}|t_{track}}$   
Set error covariance  $P_k \leftarrow P_{t_{track}|t_{track}}$   
 $\phi_k \leftarrow n_\sigma \sqrt{|P_k(1,1)|}$   
while  $\phi_k < r_{sensor}$  do  
     $\hat{X}_{k+1} \leftarrow A \hat{X}_k$   
     $P_{k+1} \leftarrow A P_k A^T + B Q B^T$   
     $\phi_k \leftarrow n_\sigma \sqrt{|P_k(1,1)|}$   
     $k \leftarrow k + 1$   
end while  
Return  $\{k\Delta t, \hat{X}_k\}$ 
```

---

words, for a one-dimensional case where the  $(1, 1)$  entry of  $P$ ,  $p_{1,1}$ , gives the variance in the  $x$  position estimation error and it is this spread that must remain smaller than the sensor footprint. In particular, taking one standard deviation in the distribution on  $\tilde{x}$  as the square root of  $p_{1,1}$ , the revisit time can be tuned such that  $n_\sigma$  standard deviations are within the sensor footprint. Given a Gaussian process in  $\hat{X}$ , the error distribution will be Gaussian, and so the normal confidence levels can be assumed, with a 99.7%, 95.4%, and 68.3% chance of detecting the target if  $n_\sigma = 3, 2, 1$ , respectively. Algorithm 4 shows the revisit calculation process, where  $(\cdot)_{t_{track}}$  is the value after tracking the target for a period,  $t_{track}$ .

The dependence on  $Q$  can be seen explicitly in the revisit algorithm, while both  $Q$  and  $R$  are implicitly considered due to the Kalman updates during the sensing interval between  $t = 0$  and  $t = t_{track}$ . Figure 5-6 shows the dependence of the revisit time on  $Q$  for various selections of  $r_{sensor}$ , and Figure 5-7 shows the dependence of revisit time on  $Q$  for varying  $R$ . Both figures show the sensitivity of the revisit time to the assumed value of  $Q$ , with a very high sensitivity for low values of  $Q$  and lower sensitivity at high values of  $Q$ . This makes sense since  $Q$  represents the “randomness” of the state evolution, and therefore larger  $Q$  means the sensor must initiate the re-track sooner since it cannot predict far enough into the future. With small  $Q$ , the sensor can have high confidence in its estimate for longer periods of time, and as  $Q$  goes to zero the

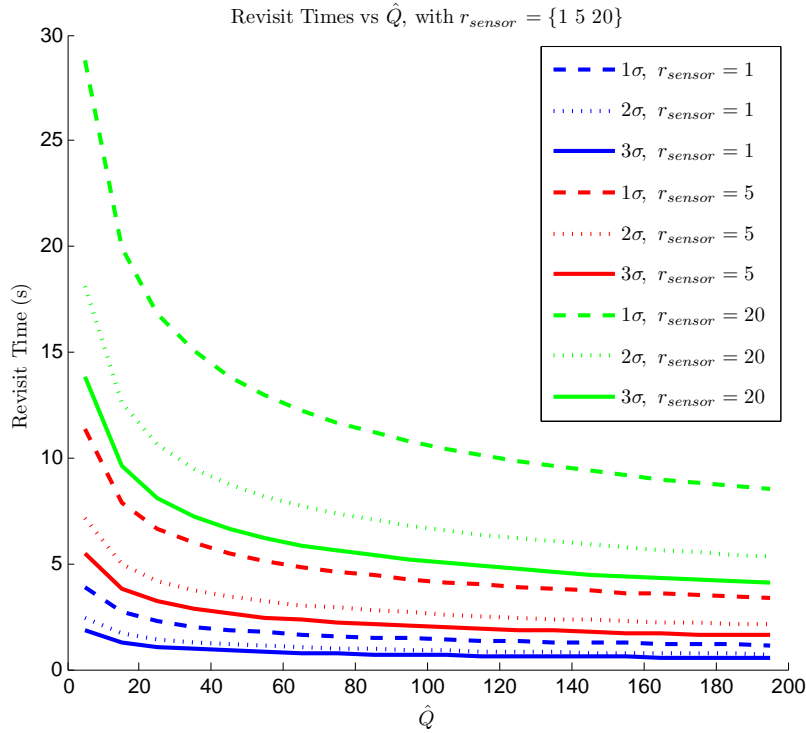


Figure 5-6: Revisit time vs  $\hat{Q}$ , parameterized by sensor size

process becomes deterministic, at which point the revisit time approaches infinity.  $R$  factors into the revisit time through the steady-state error covariance that is achieved prior to calculating the revisit. Thus, if the sensor noise is large, the steady state uncertainty is large as well, and the revisit time is decreased since it takes less time for the uncertainty ellipse to reach the size of the sensor footprint.

From Figure 5-6, motivation for determining the proper value of  $\hat{Q}$  can be reasoned as follows. Consider an initial guess for  $\hat{Q}$  of 15, a sensor size of 20, and a desired confidence of 99%, implying  $n_\sigma = 3$ . Given these conditions, the revisit algorithm will determine a revisit time around 10 seconds, with an associated estimated revisit location. If the true  $Q$  is also 15, then this confidence holds and the sensor will likely find the target. If the true  $Q$  is closer to 30, our confidence is no longer 99%, but closer to the 95% of the  $2\sigma$  curve, and the sensor would re-initiate its track 2.5 seconds after it should have to achieve the 99% level of certainty. If  $Q$  was estimated to be a value much smaller than truth and, for example, the true value is closer to 120, then the sensor will likely detect the target only 68% of the time. Conversely, if the true

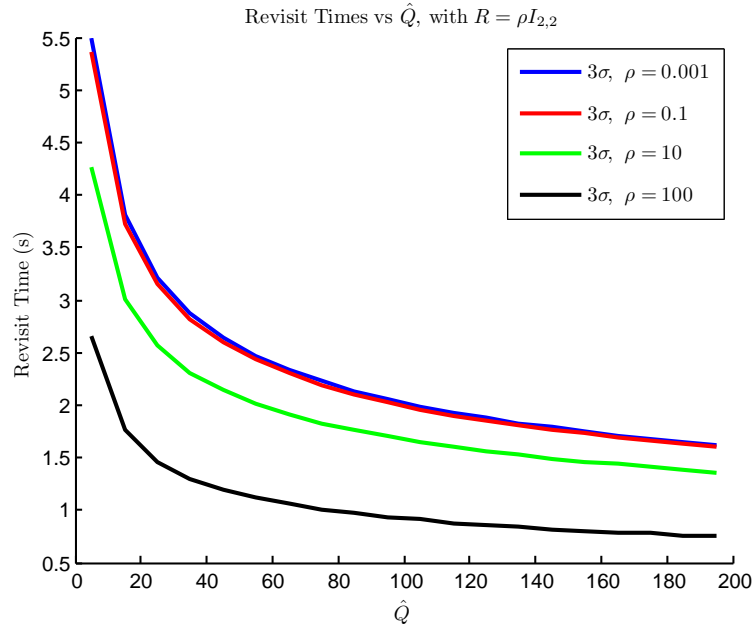


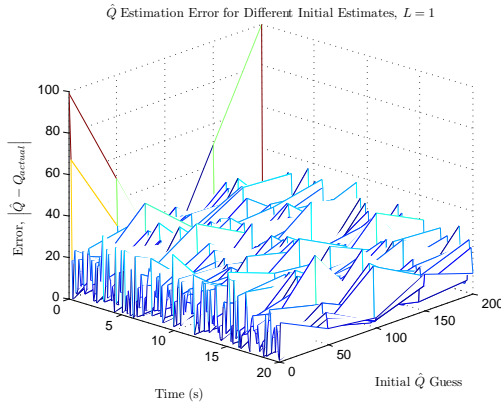
Figure 5-7: Revisit time vs  $Q$ , parameterized by sensor noise

$Q$  is smaller than the estimated  $Q$ , the sensor is more likely to detect the target, but now the re-tracking is initiated much earlier than needed. This means that the sensor doesn't take advantage of the extra time it should have had to pursue other tasks, conserve power, or do whatever action it would do that makes periodic tracking more desirable than continuous tracking.

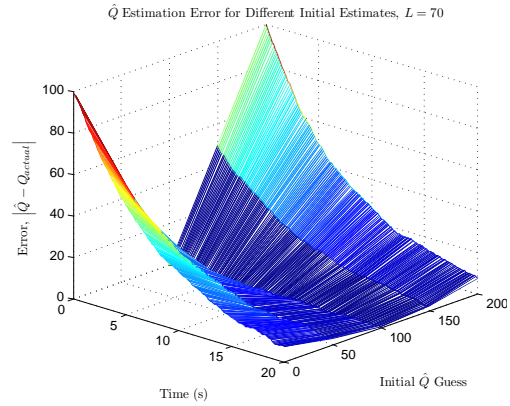
### Convergence of the Q-Adaptation Method

The use of this  $Q$ -adaptation method must first be shown to actually converge to the true value of  $Q$  from many different starting points. This brief section will show the convergence of various initial  $Q$  guesses to the true value of  $Q$ , generally corresponding to the exponential rate as suggested previously. Using a true value of  $Q_{actual} = 100$ , and initial guesses of  $\hat{Q}_0 = [1, 50, 100, 150, 200]$ , it is shown that  $\hat{Q}$  will converge to  $Q_{actual}$  given evolution according to Equations 5.9 and 5.10. Figure 5-8 shows the evolution of the error in the estimate of the process noise covariance for different values of  $L$  in Eq. 5.10. As would be expected, increasing  $L$  decreases the effective

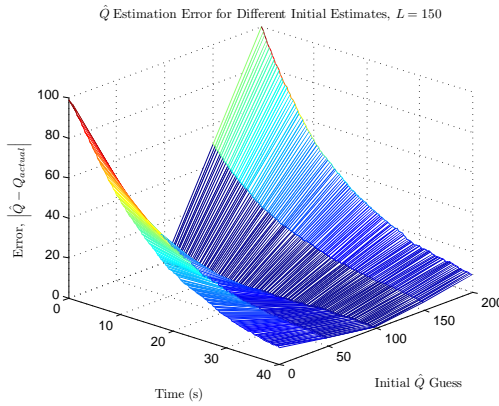




(a) Q update with  $L=1$  ( $\hat{Q}_k = \hat{Q}_k^*$ )



(b) Q update with  $L=70$



(c) Q update with  $L=150$  (note time scale)

Figure 5-8:  $\hat{Q}$  estimation trajectories with varying gains

gain on the innovation term and reduces the impact of  $\hat{Q}(t)^*$ . If  $L = 1$ , as in Figure 5-8(a), the update simply becomes  $\hat{Q}(t + 1) = \hat{Q}(t)^*$  and the process noise covariance mimics the MLE estimate for  $Q$  based on the previous time step, which results in an estimate that is quite noisy. For  $L = 150$ , as shown in Figure 5-8(c), the resulting  $\hat{Q}$  history is quite smooth and converges nicely to  $Q$ , but naturally takes longer than with smaller values of  $L$ . Intermediate values will give intermediate results, and the actual selection of  $L$  is a tuning parameter to be selected based on the given system dynamics and desired approximate settling time.

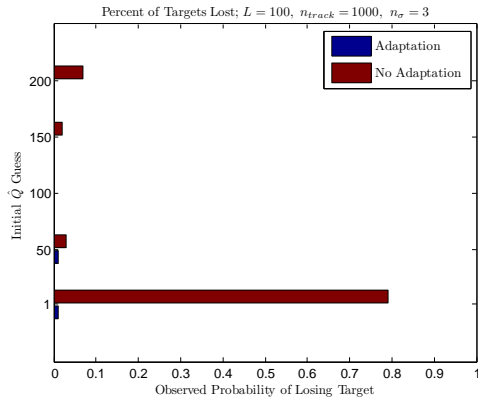
## Probability of Lost Targets

Figure 5-9 shows the percentage of targets lost after a single tracking period of  $t_{track} = 10$  seconds, where "lost" implies that the target was not within the sensor's footprint at the revisit time and location. Each run used a  $\Delta t = 0.01$ ,  $Q_{actual} = 100$ ,  $L = 100$ ,  $n_\sigma = 3, 2$ , and  $1$ , and the shown results were averaged over 100 Monte-Carlo simulations. These parameters were selected such that after  $t_{track}$  seconds the  $Q$ -adaptation process is expected to have converged to within 10% of true value, in the worst case. For consistency in each of the 100 trials, the adaptive and the non-adaptive estimations were carried out on the same true state trajectory so as to ensure no unlucky bias in favor of one or the other in terms of getting an 'easier' trajectory to follow.

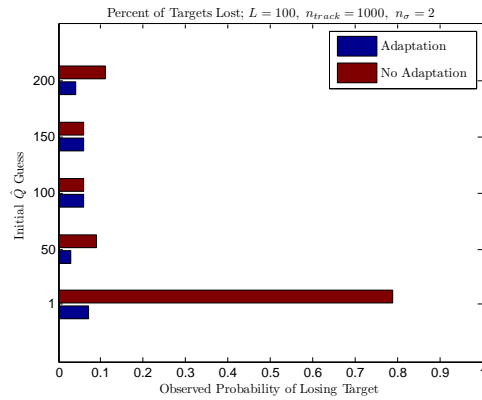
In all circumstances, the adaptive tracking method performs at least as well as non-adaptive tracking, with much better performance when the estimated  $Q$  is under the true value. For the first case with the  $3\sigma$  confidence level, shown in Figure 5-9(a), it can be seen that the adaptive method finds the target greater than 99% of the time for any tested initial  $Q$ . Particularly, the case where  $\hat{Q}_0 \ll Q$ , the non-adaptive tracking loses targets close to 80% of the time, as compared to the adaptive tracking losing only 1 or 2 percent. For the  $2\sigma$  confidence level, a similar trend holds across all initial conditions, but with all the likelihoods increased by approximately 5%. This not only continues to confirm the preference for adaptive estimation but also experimentally supports our confidence level assumption that the  $2\sigma$  multiplier leads to a confidence of approximately 95%. Finally, the  $1\sigma$  test continues both trends, with the adaptive estimation losing fewer targets than the non-adaptive at just over 30% of the targets, also matching our approximately 68% desired confidence bound.

## Observability of Process Noise

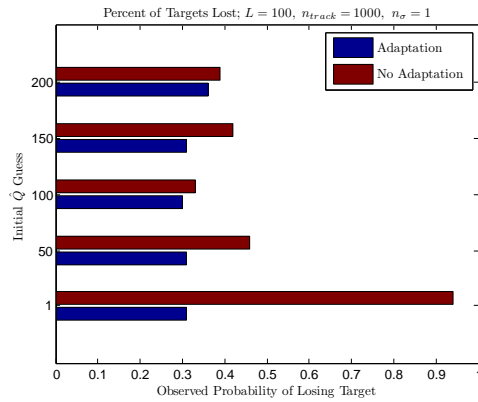
Since the process noise often factors into multiple states (in this example, the process noise is acceleration, and factors into both the position and velocity states), the best estimate of the true process noise covariance can be made when all states that the



(a)  $n_\sigma = 3$ , 99.7% Confidence



(b)  $n_\sigma = 2$ , 95.4% Confidence



(c)  $n_\sigma = 1$ , 68.3% Confidence

Figure 5-9: Percentage of targets lost at the estimated revisit time and location using adaptive and non-adaptive tracking

noise affects are measured. So far, full state feedback (FSFB) measurements have been assumed and have achieved good results. However, if only partial state feedback (PSFB) of either the position or velocity is considered instead, then the described performance can degrade. Figure 5-10 shows the convergence of  $Q$  for the same initial conditions but with different measurement capabilities. It is apparent that the position-only case does a much poorer job at converging to the true value of  $Q$ . This is possibly due to the complexity of  $Q$ 's impact on the position: At time  $t + 1$ , the position is a function of the process noise at  $t$ ,  $dt^2 \cdot w(t)$ , and the velocity at  $t$ ,  $v(t) = v(t - 1) + dt \cdot w(t - 1)$ . Thus, the apparent noise in  $x$  without having

measurements of  $v$  comes from the noise at time  $t$  and  $t - 1$ . Conversely, the  $Q$  update from taking the velocity only looks very similar to the full state feedback case<sup>3</sup>, which can likely be explained by the fact that the noise only enters the velocity update once and so produces a less ‘cluttered’ representation.

### Tracking Time-Varying $Q$ Values

One of the nice results of utilizing a dynamic estimate of  $Q$  is that it can respond to measured changes in the observed state’s noise as well as to initial errors in the  $Q$  estimate. Assuming that the new true  $Q$  value doesn’t cause the filter to diverge, the update equation can recognize this change in the system just as if it were an error akin to the initial  $Q$  guess. As with the constant  $Q$  case, the system is able to adapt to a varying  $Q$  easier if measuring the set of states with the least ‘diluted’ noise input, such as velocity-only or full state feedback. The system does still attempt to follow the changing  $Q$  with PSFB, but it takes much longer to converge.

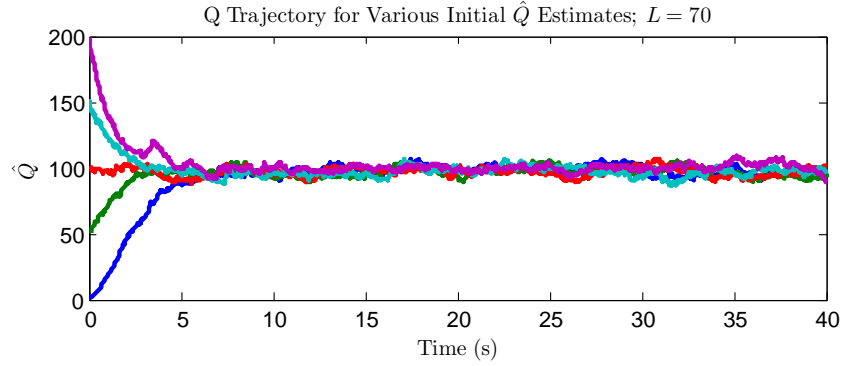
### Future Work

As mentioned in the introduction, this work contributed to the motivation for developing the hyperparameter consensus method. The specific need encountered in the adaptive tracking scenario is the question of properly sharing and updating the estimated process noise covariance matrix among a team of sensors. Though this problem has not been directly addressed in this thesis, the hyperparameter consensus method can be applied here, too. The conjugate prior to the multivariate normal distribution with known mean and unknown covariance matrix is the multivariate form of the inverse-gamma, called the inverse-Wishart distribution:

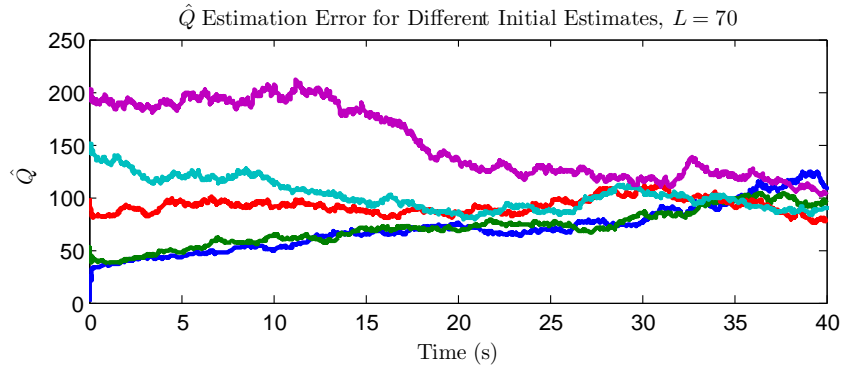
$$f_{IW}(\Sigma|\Psi, m) = \frac{|\Psi|^{m/2} |\Sigma|^{-(m+p+1)/2} e^{-\frac{1}{2}\text{trace}(\Psi\Sigma^{-1})}}{2^{mp/2} \Gamma_p(m/2)}$$

---

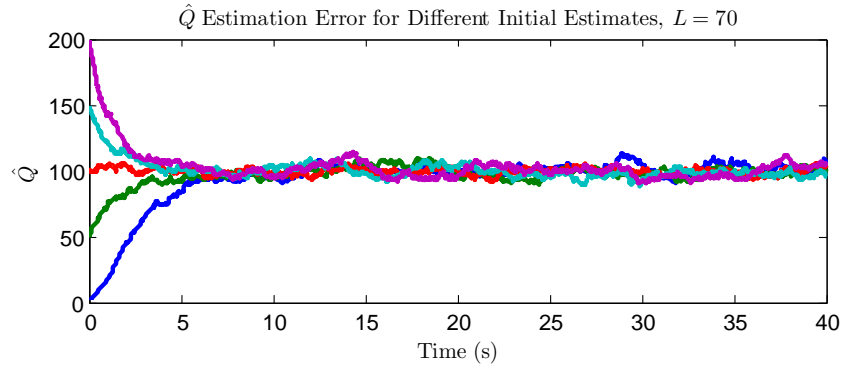
<sup>3</sup>Of course, with velocity-only measurements, there is no way to remove accumulated uncertainties in position and so the overall estimation suffers and the target is easily lost, so velocity-only measurement does not make sense in the single-sensor case.



(a) Full State Feedback Q-Adaptation



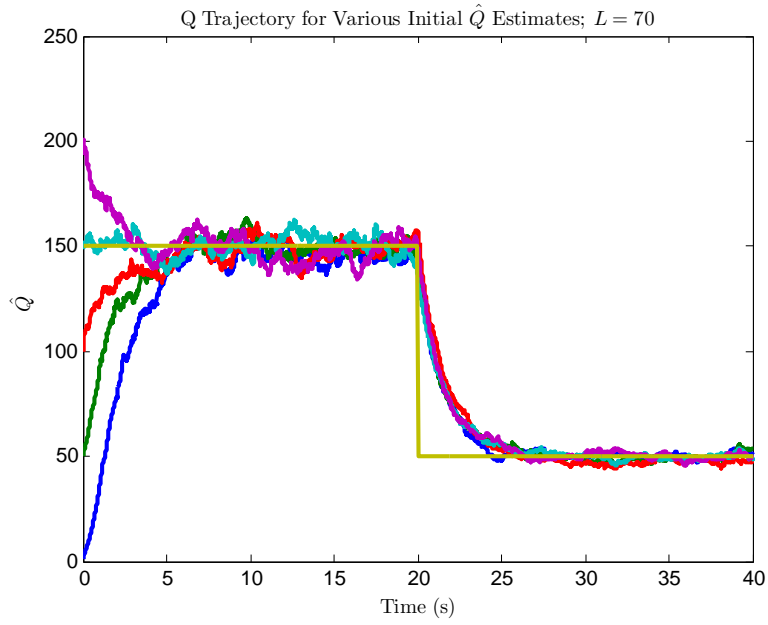
(b) Partial State Feedback Q-Adaptation - Position Only



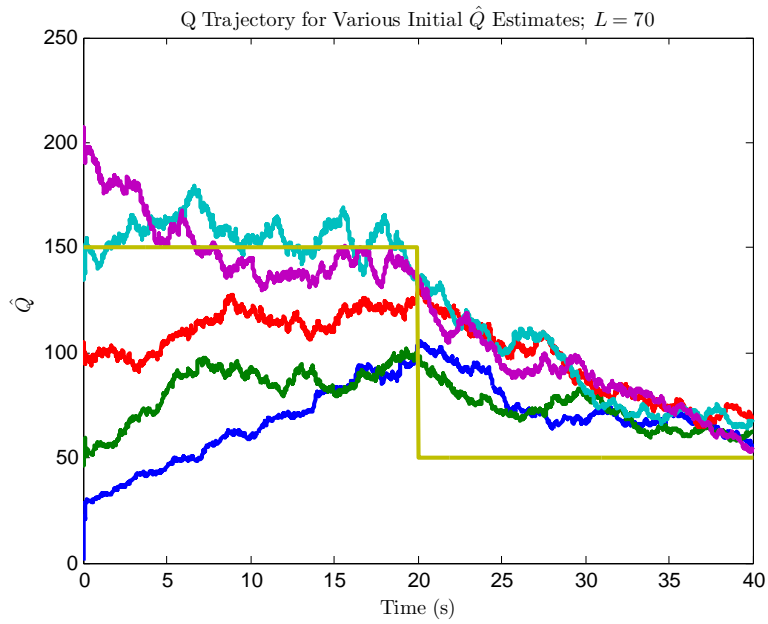
(c) Partial State Feedback Q-Adaptation - Velocity Only

Figure 5-10: Q-adaptation with different measurement capabilities

where  $\Sigma, \Psi \in \mathbb{R}^{p \times p}$  are symmetric positive definite matrices,  $m \geq p$ , and  $\Gamma_p(\cdot)$  is the multivariate gamma function. As with the previous conjugate prior distributions, the inverse-Wishart's measurement update given a multivariate observation  $z \in \mathbb{R}^{p \times 1}$  can



(a) Full State Feedback



(b) Partial State Feedback

Figure 5-11: Response to a step change in  $Q$

be simplified to an update of the hyperparameters:

$$m \leftarrow m + 1, \quad \Psi \leftarrow \Psi + zz^T$$

The estimation of covariance matrices is particularly sensitive to outliers and may require special attention to avoid divergence [87]. However, if divergence can be avoided, then the above results may be able to translate into a Bayesian estimation problem approximating the update  $zz^T$  with the immediate MLE estimate,  $Q^*$ . This would allow the agents to learn the noise characteristics of different agents over time, passing off the estimation problem to different agents on subsequent tracks.

### 5.3.2 Search Map Consensus

In addition to the Q-adaptation extension, research was conducted into Bayesian methods to aggregate uncertain probabilistic search maps. Much of the current literature on autonomous search and track assumes a centralized probability map located either at a centralized node or on each agent but updated through a fully connected network [88, 89]. In many situations either centralized planning is undesirable or a fully connected network at each time step may not be guaranteed, either of which can lead to discrepancies between the search maps contained on each distributed agent. It is important to resolve these differences in a representative manner in order to conserve as much information as possible and permit implicit coordination of the search trajectories. The ability of the Dirichlet distribution to represent both a believed probability and its confidence lends credence to its use to assist with the search map consensus problem.

#### Problem Formulation

This section will focus on the use of a counts-based representation of discrete search maps to achieve consensus representative of the initial probabilities and confidences of each of the agents. It will be assumed that the search space will be divided up into a  $n_x$  by  $n_y$  grid comprising of  $M = n_x n_y$  cells, where each cell has a probability

of containing a static target independently of any of the other cells. A group of  $N$  agents will be searching the space, each with a different sensor model defined by  $\mathbf{p}_{FP}^i$  and  $\mathbf{p}_{FN}^i$ , the probability of reading a false positive when there is no target and the probability of a false negative when there is a target. For the following discussion, it will be assumed that a sensor is able to sense only in the cell it's currently occupying. Therefore, a Bayesian probability update for sensor  $i$  given an initial probability of a target in the cell,  $\mathbf{p}^i(x, y)$ , can be derived as:

$$\mathbf{p}_{new}^i(x, y) = \begin{cases} \mathbf{p}_{there|seen}^i(x, y) & = \frac{(1 - \mathbf{p}_{FN}^i)\mathbf{p}^i(x, y)}{(1 - \mathbf{p}_{FN}^i)\mathbf{p}^i(x, y) + \mathbf{p}_{FP}^i(1 - \mathbf{p}^i(x, y))} \\ & \text{if detected} \\ \mathbf{p}_{there|not\ seen}^i(x, y) & = 1 - \frac{(1 - \mathbf{p}_{FP}^i)(1 - \mathbf{p}^i(x, y))}{1 - (1 - \mathbf{p}_{FN}^i)\mathbf{p}^i(x, y) - \mathbf{p}_{FP}^i(1 - \mathbf{p}^i(x, y))} \\ & \text{if not detected} \end{cases} \quad (5.11)$$

It is assumed the sensors are informative, such that  $0 < \mathbf{p}_{FP}, \mathbf{p}_{FN} < 0.5$ .

In previous sections, the Dirichlet and beta distributions have been shown to offer a means to estimate and agree upon uncertain probabilities. In [90], Bertuccelli introduces the use of a beta distribution over the local probabilities within each cell and uses the binomial hyperparameter update with imperfect sensors to determine how many ‘‘looks,’’ or samples from the binomial, are required to surpass a certain probability threshold within the cell. The approach taken here extends this concept by placing a local beta distribution to describe the uncertain in the probability in each cell, but bases the measurement update on the sensor model in Eq. 5.11. Therefore, given a measurement and prior probability within a cell, the sensor model depicts the desired posterior probability. This update can be considered akin to sampling repeatedly from the binomial distribution and updating the hyperparameters until the desired probability is reached. Using this approach, the hyperparameter update



is modified to

$$\begin{aligned}\alpha_a(t+1) &= \begin{cases} \alpha_a(t) + n & \text{if detected} \\ \alpha_a(t) & \text{if not detected} \end{cases} \\ \beta_a(t+1) &= \begin{cases} \beta_a(t) & \text{if detected} \\ \beta_a(t) + n & \text{if not detected} \end{cases}\end{aligned}\tag{5.12}$$

where  $n$  is the number of counts needed obtain the posterior probability, and is found by:

$$n = \begin{cases} \frac{\bar{p}_{new}^i(\alpha_i + \beta_i) - \alpha_i}{1 - \bar{p}_{new}^i} & \text{if detected} \\ \frac{(1 - \bar{p}_{new}^i)(\alpha_i + \beta_i) - \beta_i}{\bar{p}_{new}^i} & \text{if not detected} \end{cases}\tag{5.13}$$

By utilizing this update strategy, the total number of counts is automatically increased with each subsequent measurement as much as needed to obtain the desired new probability within the cell. This approach has the benefit of naturally decreasing the uncertainty of the estimate as more measurements are taken, while keeping the probability representative of the sensor model update. However, there is one caveat with this approach whereby, after repeated measurements in a single cell, the total number of counts either for or against the existence of a target can grow very large and start becoming numerically intractable. For example, as the posterior cell probability approaches 1 or 0, Eq. 5.13 states that the number of new counts required approaches infinity. Therefore, for practical considerations, it may be necessary implement an artificial ceiling on the number of counts to prevent them from approaching infinity with repeated measurements.

### Searching for Static Targets

Since this problem has an explicit Bayesian update in Eq. 5.11 that does not require the hyperparameters (unlike the MDP model estimation problem in Chapter 4), it is possible to simulate the problem without considering any explicit uncertainty in the

probabilities. In particular, it is possible to compare the evolution of the probability-based search maps versus hyperparameter-based search maps, including both local measurements and consensus stages, in order to see the long-term effect of each method. The parameter-only search problem will be considered using only the probability estimates themselves, initialized to a particular value, updated locally using Eq. 5.11, and agreed upon using average consensus. This approach will be compared to a hyperparameter method, where local measurements are incorporated using Equations 5.12 and 5.13, and the hyperparameter consensus method is used for agreement. Further, these two methods will be compared to a centralized probability map that has been initialized to the same distribution and is updated using Eq. 5.11 for every sensor at every time step.

Since the search algorithm used can produce a large effect on the overall efficiency of the search, and is not of primary interest in this research, a naïve locally greedy algorithm is implemented. After each agent has made a measurement, it either stays where it is or it moves to one of the adjacent cells. Its movement is based primarily on whichever cell is thought to have the higher probability of containing a target, though it is constrained not to enter a cell that is already occupied by another agent and preferred not to enter a cell for which the counts have reached the maximum allowed. For all neighboring cells that contain equally high probabilities, the decision is made randomly among those cells.

### **The Agreement Problem**

In the pure agreement problem to be discussed first, each agent has a different belief of the world due, for example, to different *a priori* information or due to differences in the regions searched by the vehicles prior to coming into communication range. The scenario presented in Figure 5-12 represents three agents in a map discretized into a 10 by 10 grid. Each agent begins with a shared uniform prior for the probability in each cell such that a target is equally likely to be in each cell as not. The initial search maps shown in Figures 5-12(a) through 5-12(c) are representative of each agent searching independently for a period of time prior to initiating a consensus protocol

with the other agents (the search path is overlaid in red). This delay may be due to communication constraints such as agents being out of range or that communication is only allowed periodically to conserve energy. The centralized map for the given sensor history is shown in Figure 5-12(e), with the average consensus and hyperparameter consensus results in Figures 5-12(d) and 5-12(f), respectively. As expected, the average consensus method converges to a value that is similar in shape but quite different in magnitude from the centralized estimate due to the agents considering their prior flat estimates of equal value to an estimate derived from measurements in that cell. The hyperparameter consensus method is able to properly capture the increased confidence associated with each measurement and converges to the centralized estimate.

### **The Dynamic Search Problem**

This section considers the full search problem where multiple sensors are looking for static targets in an unknown map. As mentioned, each agent can evolve and communicate their search map in either probability or hyperparameter space. This section investigates the error in the local search maps achieved using each method compared to the evolving centralized map. It will be assumed that the agents communicate their information until convergence between subsequent measurements, such that the need for concurrent consensus and measurements is avoided (though it was shown in Section 3.1.3 that hyperparameter consensus can handle this scenario if it arises). Additionally, since the consensus algorithm will be run after each measurement, when executing the hyperparameter approach, each agent maintains knowledge of the shared information from the previous consensus and adjusts their hyperparameter consensus protocol accordingly.

In order to generate a fair comparison between different network sizes, the ratio of the number of agents to the number of cells in the discretized map was kept constant at 20, such that 2 agents explored a  $5 \times 8$  map while 10 agents explored a  $10 \times 20$  map. The ratio of targets to agents was also kept approximately constant around 0.8. The scenarios were repeated 100 times each in order to determine a trend that is

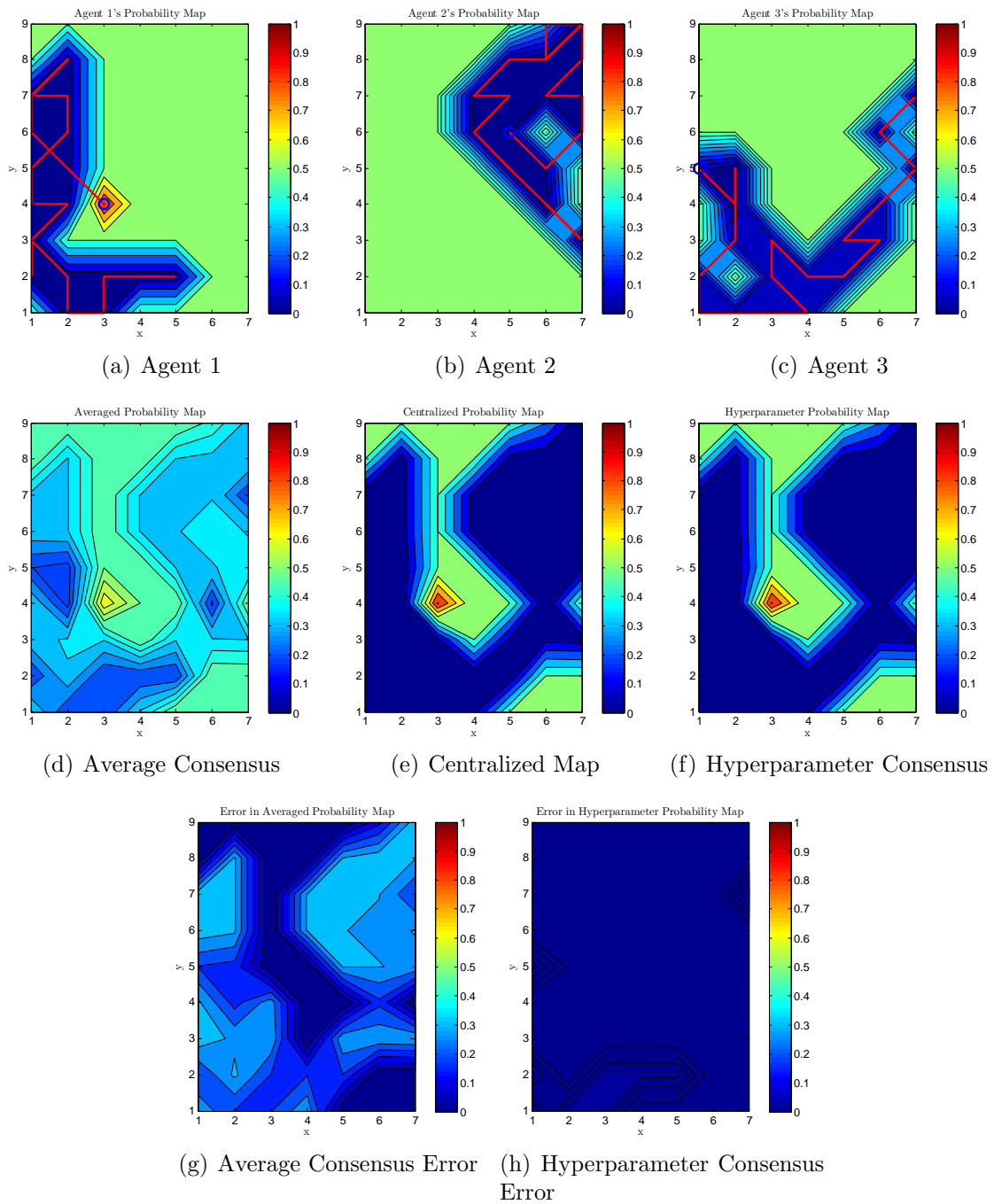


Figure 5-12: Static consensus on unique local search maps. Comparison of hyperparameter and probability-only based approaches to the centralized map

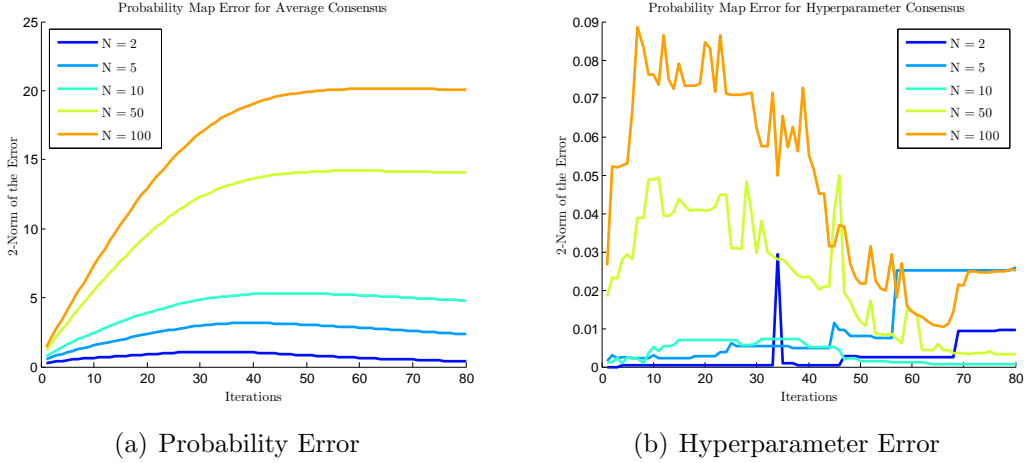


Figure 5-13: Evolution of errors in the search map using probability-based and hyperparameter-based search maps

representative of the average result. Each agent’s sensor parameters ( $\mathbf{p}_{FN}$  and  $\mathbf{p}_{FP}$ ) were given a different, random initial condition between 0 and 0.1, as well as each agent being initialized in a different location on each test. Finally, to prevent an agent from staying on a target once it has been found, it flags the cell once the probability surpasses 97% and avoids re-entering the cell for the duration of the test.

The results shown in Figures 5-13(a) and 5-13(b) represent the evolution of the error in probability belief in each cell as a function of time assuming consensus is run to convergence after each measurement. As suggested by the previous discussion, the average consensus method does not provide an accurate representation of each agent’s beliefs and so the search problem using the probability regime incurs large errors. This is especially pronounced with larger numbers of agents since each agent’s local measurements impact the post-consensus result less when there are more agents. The hyperparameter approach, on the other hand, achieves very good performance and permits the agents to each converge to the centralized map, which is updated solely according to Eq. 5.11 and without any consideration of the hyperparameters.

A trend that is apparent in the average consensus results is that the errors peak at around 40 iterations. This is because, for the ratio of agents to cells of 20 and the semi-random search algorithm, it generally takes around 40 iterations for the agents to have searched the entire map. Prior to this point, the cells that have not

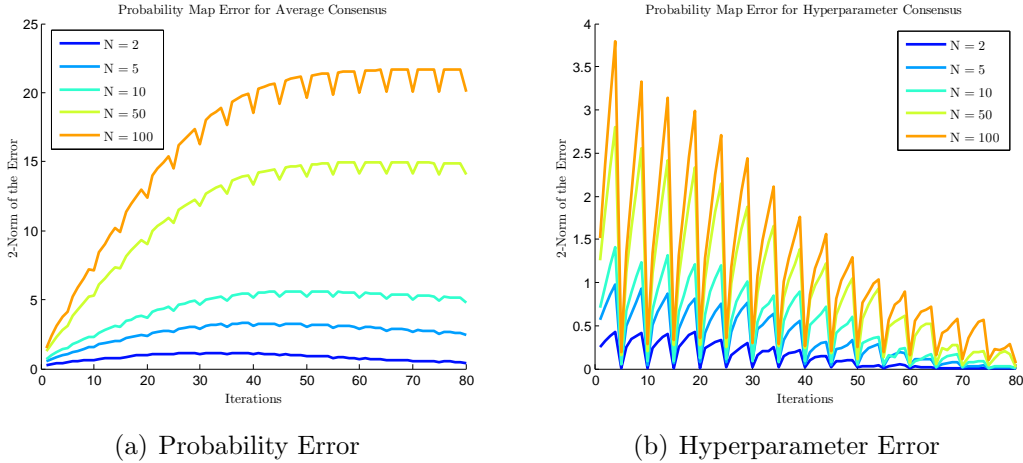


Figure 5-14: Evolution of errors in the search map using probability-based and hyperparameter-based search maps with consensus after every 5 measurements

been visited yet remain at their initial value and so do not contribute to the error, while after this point previously visited cells in which targets have not been found are revisited, causing them to begin approaching the near-zero probability of the centralized map. The hyperparameter consensus errors, on the other hand, are two orders of magnitude smaller than the average consensus errors, and lacks any really noticeable performance loss at any particular time during its execution.

If, instead of communicating after every agent takes a single local measurement, the vehicles were permitted to search independently for extended periods before running the consensus algorithm, the accuracy of the post-hyperparameter consensus map remains much higher than average consensus. Figure 5-14 shows the post-consensus error for the two consensus method when each agent observes 5 local measurements between consensus executions. In this situation, both consensus algorithms act as a correction for errors accrued between consensuses due to a lack of global knowledge of unshared, local measurements. However, only the hyperparameter consensus method is able to collapse the error to near zero after agreement, and, therefore, is able to keep the overall error in the approach an order of magnitude smaller than using average consensus.

## Future Work

The primary focus of future work on this problem is to address the inclusion of probability diffusion between cells in order to represent dynamic targets. There are three key factors at play in the application of this diffusion to the hyperparameter method. First and foremost, the probabilities should diffuse from areas of high probability to areas of low probability and should remain the same if surrounded by equal probabilities. Second, the confidence in an agent's knowledge of the probabilities in a cell should decrease over time. These imply both an evening of the counts for and against the target being in the cell, as well as a lowering of the total counts to represent the increased uncertainty. Finally, a method by which this can be done in a distributed manner needs to be addressed. This includes consistency on when and how much to scale the local counts, and how to adjust any shared information to reflect these local changes.

## 5.4 Summary

This chapter has presented a hardware implementation of a distributed Coordinated Search, Acquisition, and Track algorithm in MIT's RAVEN testbed, as well as some preliminary investigations into technologies to improve the performance of the system. The basic structure of the various modules of the CSAT architecture were presented, with emphasis on the searching and tracking algorithms in the onboard planning module. A key feature of the demonstrated system was its ability to schedule periodic tracking tasks to permit limited resources track many targets while simultaneously searching the environment. It was shown that the algorithm is able to function with the stochasticity inherent in hardware testing with three webcam-equipped quadrotors simultaneously searching the unknown environment and tracking up to five static and dynamic ground targets.

During the tests, it was noted that the revisit algorithm used to calculate the revisit times for the periodic tracking assignments was very sensitive to the estimate of the target's process noise covariance matrix. To address this sensitivity and in-

crease the reliability of the revisit calculation, research was conducted into the use of adaptive tracking techniques in the one-dimensional tracking problem to learn the process noise covariance online. This investigation confirmed that the methods proposed by [86] permit the online adaptation to the process noise, and it was shown that this adaptation greatly increased the likelihood of re-locating a target upon revisit. Specifically, the Gaussian form of the noise was utilized to estimate  $n_\sigma$  confidence bounds on the re-acquisition of the target, which were confirmed in simulation.

Finally, the implicit coordination of search paths in the CSAT algorithm motivated the investigation into methods to agree upon probabilistic search maps. This last section built upon a method introduced in [90] by utilizing a discretized search map with a beta distribution for each cell to describe the imprecise estimate of the probability that a target is in the cell. This local estimate was then updated using a customizable sensor model with allowances for false positive and false negative measurements by interpreting the posterior probability as the result of repeated samples multinomial. The hyperparameter method was compared to a pure probabilistic approach in both the static and dynamic cases, and it was again shown that the hyperparameter method maintained an accurate representation of the confidences of each agent such that each measurement was given its proper weight and the hyperparameter consensus method converged to the true centralized probability map.



# Chapter 6

## Conclusion

### 6.1 Summary

This thesis has presented a number of contributions to the consensus and cooperative control communities. Primary among these is the development of the hyperparameter consensus method to allow for multiple distributed agents to come to a proper Bayesian agreement on a parameter estimate with local uncertainties. While similar concepts have been applied in the linear-Gaussian case, the approach presented allows for agreement upon a uncertainties defined by a broader set of probability distributions which includes, but, most importantly, is not limited to the normal distribution. In particular, the hyperparameter consensus method has been shown to work for any local uncertainty model that is conjugate to a likelihood distribution, and has been demonstrated specifically for both the gamma and Dirichlet distributions. In addition to the development of the method itself, its beneficial application to distributed estimation has been shown by expediting the learning of the underlying model of a cooperative machine repair problem framed as a Markov Decision Process. Finally, the thesis concluded with an experimental implementation of a distributed and coordinated search, acquisition and track algorithm within the ACL's RAVEN testbed, which motivated a subsequent discussion of two methods to improve the performance of the algorithm in both its search and track capabilities.

Each chapter of this thesis provided a different contribution to the overall scope

of the thesis. The second chapter provided a review of current consensus techniques in the field of cooperative, coordinated control, and demonstrated a marked lack of ability to account for non-Gaussian local uncertainties in parameter estimates. In particular, three methods were discussed: average consensus; belief consensus; and Kalman consensus methods. Both average consensus and belief consensus utilized simple linear consensus protocols and have been demonstrated to have strong convergence properties over a large range of static and dynamic network topologies. However, of the three, only the Kalman filter-inspired methods accounted for any degree of uncertainty in local beliefs. Unfortunately, this uncertainty was designed for, and, therefore, only properly handled in the context of a Kalman filter problem, namely, when the local beliefs are modeled as normal distributions. Thus, the chapter concluded with an understanding that, within the current consensus community, distributed agreement with non-Gaussian uncertainties was, at best, handled improperly using Kalman consensus on the mean and variance, and, at worst, non-existent.

The third chapter presented the main contribution of this thesis: the hyperparameter consensus method. The derivation of the method rested on a number of insights from the Kalman filter and Kalman consensus approaches, including the benefits of conjugacy of distributions and the perspective on agreement as sharing pseudo-measurements from other agents. It was shown that, if a local distribution is conjugate to some likelihood function, the pseudo-measurement in Equation 3.5 representative of the local belief will be of the same form as the likelihood, and will, therefore, ensure the conjugacy of the pseudo-measurement and local uncertainty distributions. This allows for simple, closed form, additive updates of the hyperparameters, or the parameters of the local uncertainty distributions. It was shown that the desired additive updates could be achieved with a slight modification of the conventional average consensus methods presented in Chapter 2, such that convergence to the centralized hyperparameter values, and therefore the centralized Bayesian estimate, is guaranteed over any network over which linear consensus can achieve an average of initial conditions. Further, the method can account for independent, local measurements made before and during consensus, and incorporate this new infor-

mation immediately into the network to guide the consensus to the resulting, new centralized estimate.

Some primary results of the method are outlined as follows:

- Guaranteed convergence over arbitrary, known networks that are strongly connected. This powerful result states that the agents can come to agreement on the Bayesian fused estimate of a parameter over a network with any structure, sparse or fully connected, so long as it is at least strongly connected and the consensus eigenvector,  $\nu$ , is known. This is a very important result as it distinguishes this method from existing techniques within the Data Fusion community, which seek to achieve the same end as hyperparameter consensus, but which often require prohibitively complex channel filters and message-passing schemes to avoid network-induced redundancies. By using a protocol based on a linear consensus algorithm, and through weighting the initial conditions dependent on the value of  $\nu$ , these redundancies are avoided. Further, since the weights are applied to the hyperparameters and not the parameter of interest, itself, the hyperparameter ratio defining the MMSE parameter estimate is unchanged, meaning that the best estimate of the parameter value remains representative of the currently available information during the process of agreement.
- Improved accuracy of hyperparameter consensus versus parameter consensus over unknown networks. It was demonstrated that, even if the network, and therefore consensus eigenvector, is not known, then running a consensus protocol on the hyperparameters (where the protocol was designed to give an average consensus value on a balanced network) was shown to provide a more accurate and more precise steady-state parameter estimate compared to the centralized Bayesian estimate, versus the same algorithm running on the best local estimate of the parameter itself. This result was achieved over all tested, connected network topologies and ranges of network-induced biases. Therefore, even in situations where the network topology is unknown, hyperparameter consensus

is able to capture enough of the local uncertainty to bias the results towards the proper value.

- Shared prior information can be easily accounted for. The general concept of shared information, which drives the necessity for channel filtering in data fusion, is simply handled using hyperparameter consensus. Since network-induced biases are accounted for by local weightings, shared information in the hyperparameter setting takes the form of common priors or results from previously converged consensus. Fortunately, this information is shared globally throughout the network, and, therefore, does not require individual treatment between neighbors. Instead, the agents simply either do not weight this prior information before consensus, or subtract the representative hyperparameters from their own and come to consensus on the difference. This permits repeated estimation and consensus intervals without collapsing the variance in the beliefs due to repeated confirmation of the same information.
- Convergence properties of the algorithm are already very well known and studied. Since hyperparameter consensus is an extension of traditional linear consensus methods to a set of variables effectively once removed from parameter of interest, the convergence properties of the algorithm rely only on the convergence properties of linear, average consensus itself. This field has been studied immensely over the last decade, in terms of both existence of an equilibrium and speed of convergence to that value, over a broad range of static and dynamic networks. In particular, consensus in terms of flocking with time-varying nearest-neighbor networks [20], small world networks and convergence speeds [27], and time delays [23]. See Sections 1.1.1 and 2.3.1 for references.
- Allows for distributed agreement on any parameterized distribution in the exponential family as well as some non-exponential distributions. This result is due to the fact that a conjugate prior exists for every distribution in the exponential family (*i.e.*, normal, Poisson, exponential, gamma etc.), and that conjugate priors exist for a number of non-exponential distributions, too (*i.e.* multino-

mial). Since the conjugate prior defines the uncertainty of the parameters of the likelihood distribution, and since it has been shown that hyperparameter consensus converges on the centralized Bayesian estimate, then it follows that any distributions that are parameterized by the value being agreed upon also become aligned. This has important implications in distributed estimation as, with time, a distributed sensor network running hyperparameter consensus can achieve agreement on an underlying stochastic process, itself, when the process is modeled by one of the relevant distributions.

Chapter 4 introduced the hyperparameter method to a distributed estimation problem in the context of model-based learning in a Markov Decision Process setting. The application was formulated as a multi-machine repair problem where multiple agents observed independent, identically distributed stochastic processes; namely, transitions of each machine's state due to different actions. It was demonstrated that the hyperparameter consensus method effectively aggregated each agent's individual observations, rather than just its current model estimate, such that larger networks were able to learn much faster than individual agents or the same network running average consensus on the model parameters. This increased learning speed naturally led to performance improvement by significantly reducing the learning transient and allowing each agent to locally solve the MDP using near-truth transition probabilities (based on a Dirichlet uncertainty model) to achieve near-optimal solutions. The problem was also extended to a Markov-process inspired form where each action has a stochastic execution time associated with it. Similarly improved results were achieved in this situation, where each agent was simultaneously estimating an exponential distribution on the time required for each action. Though these results are also theoretically achievable using data fusion techniques over simple networks, the application to up to 200 nodes over arbitrary networks would require significantly complicated if not intractable messaging schemes and, in a general sense, would be impractical with current data fusion techniques.

The last chapter presented a different aspect of the coordinated control problem through the implementation of a coordinated search, acquisition, and track (CSAT)

algorithm in MIT’s RAVEN testbed. The decentralized algorithm was described with a focus on the synergistic combination of the competing search and track tasks, and, in particular, how each was allocated among the different assets of the fleet through the use of periodic tracking of known targets. Though using simple, commercial vehicle hardware with minimal computational capacity, this decentralized nature was maintained throughout the implementation by the execution of each vehicle’s control modules on its dedicated ground computer. Each of the quadrotor aircraft composing the fleet was outfitted with a camera to permit detection of independently controlled ground vehicles using vision in the loop.

The experimental tests demonstrated the robustness of the system to the uncertainties implicit in hardware tests, such as environment and sensor noise, and the ability of the algorithm to successfully trade-off between the competing search and track objectives. However, it was also shown that this trade-off is sensitive to the assumed model of the targets to be tracked. In order to mitigate any performance loss induced by a poor knowledge of the tracked vehicle’s command inputs (modeled as process noise), an adaptive tracking algorithm was tested in a one-dimensional tracking scenario. It was shown that not only does the algorithm converge to the true, observed process noise, but that, in doing so, it mitigated the possibility of losing a target upon a scheduled revisit.

A second improvement was also investigated, though focused on the search problem. Specifically, in decentralized networks with intermittent communication, it is very possible for individual probabilistic search maps updated locally by each agent to deviate from each other, and so search map fusion based on hyperparameter consensus was investigated to solve the problem. It was shown that, under the search map representation and sensor model used, hyperparameter consensus on the uncertain probabilities within each cell of a discretized search map allowed the agents to converge to an equivalent centralized search map with little to no appreciable error. Further, this approach highlighted the ability of repeated measurement and hyperparameter consensus phases to disseminate newly acquired information across a network without incurring error or the need for channel filtering.

## 6.2 Future Work

Much of the future work applicable to the hyperparameter consensus method falls under one of two categories: further expanding the range of applicable distributions and estimation of non-static processes. The first concept requires a deeper investigation of the form and generation of more generic conjugate priors as is presented in [91] and references therein. Applications in this thesis have been confined to common conjugate pairs (ie. normal-normal, exponential-gamma, multinomial-Dirichlet), though the concept of conjugacy has a much richer theoretical basis than represented here. It may be possible to extend the concepts proposed for use in the hyperparameter consensus method to a more generic class of distributions through use of the mean conjugate prior proposed in [91].

The second avenue of investigation into distributed estimation of non-static processes using hyperparameter consensus is an extension that is critical to many proposed applications. This line of research is akin to the addition of the state propagation step between subsequent measurements in the Kalman filter, which adds much complexity in the distributed setting. Particular questions that arise are how to handle out-of-order or otherwise temporally delayed measurements (a problem avoided in this thesis due to the assumption of a static process, in which order does not matter) and the specific impact of the propagation on the hyperparameters themselves (which is sufficiently known in the Kalman sense already, but may be more complicated in other settings). The first focus is an ongoing area of research in distributed tracking [92, 93], and is often assumed avoidable by using scheduled communication and assuming convergence between subsequent propagation/measurement phases [34, 40]. However, especially when using consensus algorithms, the convergence is not instantaneous, and for any high-bandwidth sensor the measurement rate may be sufficiently fast to prevent consensus between measurements. Therefore, in this application as in the rest of the community, it remains a necessary focus to determine how to update out-of-order or delayed measurements. The solution of this problem may be directly linked with the second presented question of how the state propagation updates affect

the hyperparameters themselves. In the Kalman filter framework, this update of  $\hat{x}$  and  $P$  is done via the well-known propagation equations. In other frameworks, such as with the Dirichlet prior, the dynamics update may be significantly more complicated and restrictive than in the Kalman setting. For example, the reason that the Kalman prediction equations are in closed form are because the prior distribution is also conjugate to the propagation model (in particular, the process noise), which affords equally simple hyperparameter updates. The addition of an equivalent process noise to a probability vector in the Dirichlet case is not so well defined. Scaling the hyperparameter counts by a factor may be the best approach (see [54]), though it does not correspond to a multinomial update as a parallel to Kalman filtering may suggest. Further, in the problem of repeated estimation (as in the search map application of Chapter 5), any propagation model may complicate the knowledge of shared information. For example, if the probability of one cell of a search map diffuses into a neighboring cell, how much, if any, of the shared information travels with it and how is it accounted for in the subsequent consensus problem? What about if communication is sparse and multiple agents have diffused probabilities incorrectly due to insufficient knowledge of the other agents' maps?



# Appendix A

## Bayesian Derivation of the Kalman Filter

The following derivation is adapted from [46]. The Bayesian formulation of the Kalman Filter utilizes two key properties prior to making any distinction on the distribution of the random variables involved. First is the Markov property, whereby the combined information available from a history of states,  $\{x_0, \dots, x_n\}$ , or measurements,  $\{z_0, \dots, z_n\}$ , is entirely encapsulated by the most recent state,  $x_n$ , or measurement,  $z_n$ . The second property is that of conditional independence, where  $p(x_{k+1}|x_k)$  is independent of  $p(x_{j+1}|x_j)$  for all  $j \neq k$ , and  $p(z_k|x_k)$  is independent of everything else, except for  $p(z_j|k_j)$  when  $j = k$ . Using these properties and knowledge of  $p(x_{k+1}|x_k)$  and  $p(z_k|x_k) \forall k$  from the update and measurement equations

$$x_{k+1} = f(x_k, w_k, t_k, t_{k+1}) \quad E[w_k w_j^T] = 0 \text{ if } k \neq j \quad (\text{A.1})$$

and

$$z_k = g(x_k, v_k, t_k) \quad E[v_k v_j^T] = 0 \text{ if } k \neq j \quad (\text{A.2})$$

we can derive the Bayesian update for  $p(x_{n+1}|Z_n = \{z_1, \dots, z_n\})$  as:

$$\begin{aligned}
p(x_{n+1}|Z_n) &= \int_{x_0, \dots, x_n} p(x_0, \dots, x_n, x_{n+1}|Z_n) dx_0 \cdots dx_n \\
&= \int_{x_0, \dots, x_n} p(x_0, \dots, x_n, x_{n+1}) p(Z_n|x_0, \dots, x_n, x_{n+1}) / p(Z_n) dx_0 \cdots dx_n \\
&\propto \int_{x_0, \dots, x_n} p(x_{n+1}|x_0, \dots, x_n) p(x_0, \dots, x_n) p(Z_n|x_0, \dots, x_n, x_{n+1}) dx_0 \cdots dx_n \\
&= \int_{x_0, \dots, x_n} p(x_{n+1}|x_n) p(x_0, \dots, x_n) p(Z_n|x_0, \dots, x_n, x_{n+1}) dx_0 \cdots dx_n \\
&= \int_{x_0, \dots, x_n} p(x_{n+1}|x_n) p(x_n|x_0, \dots, x_{n-1}) p(x_0, \dots, x_{n-1}) \\
&\quad \times p(Z_n|x_0, \dots, x_n, x_{n+1}) dx_0 \cdots dx_n \\
&\quad \vdots \\
&= \int_{x_0, \dots, x_n} p(x_{n+1}|x_n) \prod_{i=1}^n p(x_i|x_{i-1}) p(x_0) p(Z_n|x_0, \dots, x_n, x_{n+1}) dx_0 \cdots dx_n \\
&= \int_{x_0, \dots, x_n} p(x_{n+1}|x_n) \left( \prod_{i=1}^n p(x_i|x_{i-1}) \right) p(x_0) \left( \prod_{i=1}^n p(z_i|x_i) \right) dx_0 \cdots dx_n \\
&= \int_{x_0, \dots, x_n} p(x_{n+1}|x_n) \left( \prod_{i=1}^n p(z_i|x_i) p(x_i|x_{i-1}) \right) p(x_0) dx_0 \cdots dx_n \\
&= \int_{x_n} dx_n \left( \int_{x_0, \dots, x_{n-1}} p(x_n|x_{n-1}) \left( \prod_{i=1}^{n-1} p(z_i|x_i) p(x_i|x_{i-1}) \right) p(x_0) \right) \\
&\quad \times p(x_{n+1}|x_n) p(z_n|x_n) dx_0 \cdots dx_{n-1} \\
&= \int_{x_n} p(x_n|Z_{n-1}) p(x_{n+1}|x_n) p(z_n|x_n) dx_n \tag{A.3}
\end{aligned}$$

The above can be modified into two recursive functions:

$$\begin{aligned}
\text{Predict: } \underbrace{p(x_{k+1}|Z_k)}_1 &= \int dx_k p(x_{k+1}, x_k|Z_k) \\
&= \int p(x_{k+1}|x_k, Z_k) p(x_k|Z_k) dx_k \\
&= \int p(x_{k+1}|x_k) \underbrace{p(x_k|Z_k)}_2 dx_k
\end{aligned}$$

and

$$\begin{aligned}
\text{Update: } \underbrace{p(x_k|Z_k)}_2 &= p(x_k|Z_{k-1}, z_k) \\
&\propto p(z_k|x_k, Z_{k-1})p(x_k|Z_{k-1}) \\
&= p(z_k|x_k) \underbrace{p(x_k|Z_{k-1})}_1
\end{aligned}$$

Up to this point, we have made no specification of the types of distributions used for any of these random variables except that the ‘noises’  $w_k$  and  $v_k$  are uncorrelated (and hence that the propagation and measurements are conditionally independent), and that the state follows the Markov property. Given this, the random variables  $x_k$ ,  $z_k$ ,  $w_k$ , and  $v_k$  could be derived from any possible distribution that satisfy the aforementioned constraints and update equations.

In the framework of a Bayesian likelihood update, Equation A.3 uses the prior  $p(x_k|Z_{k-1})$ , likelihood function  $p(z_k|x_k)$ , and posterior  $p(x_k|Z_k) \propto p(x_n|Z_{n-1})p(z_n|x_n)$ . This posterior is then propagated to a new prior estimate  $p(x_{k+1}|Z_k)$  by integrating the posterior state estimate times the transition probability across all possible values of the intermediate state  $x_n$  (the current derivation effectively calculates the update first, then addresses the propagation step).

Let  $G(x, \Sigma, \mu) = G(\mu, \Sigma, x)$  be the Gaussian:

$$\frac{1}{(2\pi)^{n/2}(\det[\Sigma])^{1/2}} e^{\frac{1}{2}[x-\mu]^T \Sigma^{-1}[x-\mu]}$$

where  $n$  is the dimension of  $x$ , and all matrices are of such dimension that the above is a valid statement. Reference [94] shows that the product of two particular Gaussian functions can be given by:

$$\begin{aligned}
G(x, P, \mu)G(z, R, Hx) &= G(x, \Lambda, \lambda)G(z, R + HPH^T, H\mu) \\
\Lambda &= (H^T R^{-1}H + P^{-1})^{-1} \\
\lambda &= \mu + (H^T R^{-1}H + P^{-1})^{-1} H^T R^{-1}(z - H\mu)
\end{aligned}$$

If we let the noises  $w_k$  and  $v_k$  be normally distributed with mean 0 and covariance  $Q_k$  and  $R_k$ , respectively, then Equations A.1 and A.2 define the random variables  $x_{k+1}$  and  $z_k$  in terms of these noises. Additionally, if we assume linear dynamics in the propagation and measurement functions of the form  $f(x_k, w_k, t_k, t_{k+1}) = \Phi(t_{k+1}, t_k)x_k + \Gamma_k w_k$  and  $g(x_k, v_k, t_k) = H_k x_k + v_k$ , then we observe that

$$\begin{aligned} x_{k+1}|x_k &\sim \mathcal{N}(\Phi(t_{k+1}, t_k)x_k, \Gamma_k Q_k \Gamma_k^T) \\ z_k|x_k &\sim \mathcal{N}(H_k x_k, R_k) \end{aligned}$$

where  $\Phi(t_{k+1}, t_k)$  will henceforth be denoted  $A_k$ , as is commonly done, and signifies the deterministic portion of the state transformation. From these, we get expressions for some of the probabilities used earlier as:

$$\begin{aligned} p(x_{k+1}|x_k) &= G(x_{k+1}, \Gamma_k Q_k \Gamma_k^T, A_k x_k) \\ p(z_k|x_k) &= G(z_k, R_k, H_k x_k) \end{aligned}$$

Defining  $z_0 = \emptyset \rightarrow Z_0 = \emptyset$ , we let  $p(\cdot|Z_0) = p(\cdot|z_0) = p(\cdot)$ . Finally, let the initial estimate of  $x$  be  $\hat{x}_{0|0}$ , with an associated covariance  $P_{0|0}$ . The Bayesian equations above can then be written as:

$$\begin{aligned} \text{Predict: } p(x_1|Z_0) &= \int dx_0 p(x_1|x_0)p(x_0|Z_0) \\ &= \int p(x_1|x_0)p(x_0)dx_0 \\ &= \int G(x_1, \Gamma_0 Q_0 \Gamma_0^T, A_0 x_0) G(x_0, P_{0|0}, \hat{x}_{0|0}) dx_0 \\ &= G(x_1, \Gamma_0 Q_0 \Gamma_0^T + A_0 P_{0|0} A_0^T, A_0 \hat{x}_{0|0}) \\ &= G(x_1, P_{1|0}, \hat{x}_{1|0}) \end{aligned}$$

$$\begin{aligned}
\text{Update: } p(x_1|Z_1 = \{z_1\}) &= p(z_1|x_1)p(x_1|Z_0) \\
&= G(z_1, R_1, H_1x_1)G(x_1, P_{1|0}, \hat{x}_{1|0}) \\
&= G(\hat{x}_{1|1}, P_{1|1}, x_1)G(z_1, R_1 + H_1P_{1|0}H_1^T, H_1\hat{x}_{1|0}) \\
&\propto G(x_1, P_{1|1}, \hat{x}_{1|1})
\end{aligned}$$

Where we turn to proportionality at the end since the dropped term is constant and defined as the value required to normalize the remaining distribution. Finally, we have introduced  $\hat{x}_{1|1}$  and  $P_{1|1}$ , which are found in the general case as:

$$\begin{aligned}
P_{k|k} &= (H_k^T R_k^{-1} H_k + P_{k|k-1}^{-1})^{-1} \\
\hat{x}_{k+1|k+1} &= A_k \hat{x}_{k|k} + (H_{k+1}^T R_{k+1}^{-1} H_{k+1} + P_{k+1|k}^{-1})^{-1} H_{k+1}^T R_{k+1}^{-1} (z_{k+1} - H_{k+1} A_k \hat{x}_{k|k}) \\
\hat{x}_{k+1|k+1} &= \hat{x}_{k+1|k} + P_{k+1|k+1} H_{k+1}^T R_{k+1}^{-1} (z_{k+1} - H_{k+1} \hat{x}_{k+1|k})
\end{aligned}$$

where, additionally,  $P_{k+1|k+1} H_{k+1}^T R_{k+1}^{-1}$  is called the Kalman gain,  $K_k$ . Thus, we have arrived at the common update equations of the Kalman Filter, while the prediction equations are given by:

$$\begin{aligned}
P_{k+1|k} &= A_k P_{k|k} A_k^T + \Gamma_k Q_k \Gamma_k^T \\
\hat{x}_{k+1|k} &= A_k \hat{x}_{k|k}
\end{aligned}$$

In notation similar to that used for the covariance, the mean of  $x_k|Z_{k-1}$ , is written as  $\hat{x}_{k|k-1}$ , and similarly  $E[x_k|Z_k]$  is denoted  $\hat{x}_{k|k}$ .

In the extension to the multi-sensor case, consider now  $N_s$  total sensors, of which only  $m$  sensors  $i = \{I_k(1), \dots, I_k(m)\}$  are active at time  $t$ . Each sensor has its own measurement equation, given by  $z_k^i = g(x_k, v_k^i, t_k) \quad E[v_k^i (v_l^j)^T] = R_k^i \delta(t_k - t_l) \delta(i - j)$  (ie. all sensing noises are i.i.d. and uncorrelated in time as well as between sensors). If, in the above derivation, we let  $z_k$  denote the set of all measurements made at time  $k$ , such that  $z_k = \{z_k^i\} \forall i \in I_k$ , and utilizing our assumption of conditional independence through uncorrelated white noise, we can obtain a new measurement proba-

bility  $p(z_k|x_k) = \prod_{i \in I_k} p(z_k^i|x_k)$ , where now each  $p(z_k^i|x_k) \sim N(H_k^i x_k, R_k^i)$ . Thus, the centralized update equations will change to:

$$\begin{aligned}
p(x_k|Z_k = \{z_1, \dots, z_k\}) &= p(z_k|x_k)p(x_k|Z_{k-1}) \\
&= \prod_{i \in I_k} p(z_k^i|x_k)p(x_k|Z_{k-1}) \\
&= \prod_{i \in I_k} G(z_k^i, R_k^i, H_k^i x_k)G(x_k, P_{k|k-1}, \hat{x}_{k|k-1}) \\
&= G \left( \begin{bmatrix} z_k^{I_k(1)} \\ \vdots \\ z_k^{I_k(m)} \end{bmatrix}, \begin{bmatrix} R_k^{I_k(1)} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & R_k^{I_k(m)} \end{bmatrix}, \begin{bmatrix} H_k^{I_k(1)} \\ \vdots \\ H_k^{I_k(m)} \end{bmatrix} x_k \right) \\
&\quad \times G(x_k, P_{k|k-1}, \hat{x}_{k|k-1}) \\
&= G(\mathbf{z}_k, \mathbf{R}_k, \mathbf{H}_k x_k)G(x_k, P_{k|k-1}, \hat{x}_{k|k-1}) \\
&= G(x_k, P_{k|k}, \hat{x}_{k|k})G(\mathbf{z}_k, \mathbf{R}_k + \mathbf{H}_k P_{k|k-1} \mathbf{H}_k^T, \mathbf{H}_k \hat{x}_{k|k-1}) \\
&\propto G(x_k, P_{k|k}, \hat{x}_{k|k})
\end{aligned}$$

where  $P_{k|k}$  and  $\hat{x}_{k|k}$  are now given by:

$$\begin{aligned}
P_{k|k} &= \left( P_{k|k-1}^{-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k \right)^{-1} \\
&= \left( P_{k|k-1}^{-1} + \begin{bmatrix} H_k^{I_k(1)} \\ \vdots \\ H_k^{I_k(m)} \end{bmatrix}^T \begin{bmatrix} R_k^{I_k(1)-1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & R_k^{I_k(m)-1} \end{bmatrix} \begin{bmatrix} H_k^{I_k(1)} \\ \vdots \\ H_k^{I_k(m)} \end{bmatrix} \right)^{-1} \\
&= \left( P_{k|k-1}^{-1} + \sum_{i \in I_k} H_k^{i^T} R_k^{i-1} H_k^i \right)^{-1} \\
&= \left( P_{k|k-1}^{-1} + \sum_{i \in I_k} \mathbf{I}_k^i \right)^{-1} \tag{A.4}
\end{aligned}$$

$$\begin{aligned}
\hat{x}_{k|k} &= \hat{x}_{k|k-1} + P_{k|k} \mathbf{H}_k^T \mathbf{R}_k^{-1} (\mathbf{z}_k - \mathbf{H}_k \hat{x}_{k|k-1}) \\
&= P_{k|k} \left( P_{k|k}^{-1} \hat{x}_{k|k-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{z}_k - \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k \hat{x}_{k|k-1} \right) \\
&= P_{k|k} \left( \left( P_{k|k}^{-1} - \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k \right) \hat{x}_{k|k-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{z}_k \right) \\
&= P_{k|k} \left( P_{k|k-1}^{-1} \hat{x}_{k|k-1} + \sum_{i \in I_k} H_k^{i,T} R_k^{i-1} z_k^i \right) \\
&= P_{k|k} \left( P_{k|k-1}^{-1} \hat{x}_{k|k-1} + \sum_{i \in I_k} \mathbf{i}_k^i \right) \tag{A.5}
\end{aligned}$$

If we let  $Y_{k|k} = P_{k|k}^{-1}$  and  $y_{k|k} = Y_{k|k} \hat{x}_{k|k}$ , then Equations A.4 and A.5 become:

$$\begin{aligned}
Y_{k|k} &= Y_{k|k-1} + \sum_{i \in I_k} \mathbf{I}_k^i \\
y_{k|k} &= y_{k|k-1} + \sum_{i \in I_k} \mathbf{i}_k^i
\end{aligned}$$

The above equations are representative of the update equations for the Information Filter, where  $\mathbf{I}$  and  $\mathbf{i}$  are the measurement covariance matrix and measurement vector, respectively. The Information Filter is used as the framework for both the distributed Kalman filter [34] and the Kalman consensus filter [4], where both assume that, in place of an explicit measurement equation, the update step is obtained through the aggregation of the state estimates of the various agents.

## A.1 Distributed Kalman Filter

The distributed Kalman filter performs a local Bayesian state and covariance update with local information, then augments the local estimates with information from the other sensors. This arrangement assumes a fully connected network among all sensing nodes. The transmitted information is equivalent to the measurement vector, but rearranged as  $\mathbf{i}_{k+1}^j = \tilde{P}_{k+1|k+1}^{j-1} \tilde{x}_{k+1|k+1}^j - P_{k+1|k}^{j-1} \hat{x}_{k+1|k}^j$ , where  $\tilde{(\cdot)}$  is the local updated value, and the remaining terms are the propagated fused values from the previous iteration. Similarly, the covariance update is obtained with a transformation of the  $\mathbf{I}$

matrix to  $\mathbf{I}_k^j = \tilde{P}_{k+1|k+1}^{j-1} - P_{k+1|k}^{j-1}$ . This requires that the measurements are all made synchronously and reported with no communication delays.

## A.2 Kalman Consensus Filter

The Kalman consensus filter is used as a means by which multiple agents can come to agreement on a parameter that is assumed to be the mean of a normal distribution with known covariance. This assumption permits a conjugate prior normal distribution that defines a belief on the parameter through an estimated mean and covariance. In other words, each agent is trying to come to agreement on the value  $x$ , and each agent has an associated estimate of the value,  $\mu_{x_k}^i$ , and a covariance in the estimate,  $\Sigma_{x_k}^i$ . The outcome of the filter is the mean of a Gaussian that obtains the highest *a posteriori* estimate of  $x$  given measurements  $\mu_{x_k}^i$  with covariances  $\Sigma_{x_k}^i$ , which is denoted  $x_k^*$ . This value is assumed to have trivial dynamics but perturbed by process noise, such that  $x_{k+1}^* = x_k^* + w_k$ ,  $w_k \sim \mathcal{N}(0, Q_k)$ . If we let  $z_k^i = x_k^* + (\mu_{x_k}^i - x_k^* + \nu_k^{ij}) = x_k^* + v_k^i$ , where  $\nu_k^{ij} \sim \mathcal{N}(0, \Omega_k^{ij})$ , we can define  $H_k^i = \mathbf{1}$  and  $E[v_k^i v_k^{iT}] = R_k^i = \text{diag}(P_k^i + \Omega_k^{ij})$  where  $P_k^i = E[(\mu_{x_k}^i - x_k^*)(\mu_{x_k}^i - x_k^*)^T]$ . Subbing these values into the previous equations leads to the filter.



# Bibliography

- [1] W. Ren, R. W. Beard, and T. W. McLain, *Coordination Variables and Consensus Building in Multiple Vehicle Systems*. Springer-Verlag, 2004, pp. 171–188.
- [2] W. Ren, R. W. Beard, and E. M. Atkins, “Information consensus in multivehicle cooperative control,” *IEEE Control System Magazine*, vol. 27, no. 2, pp. 71–82, April 2007.
- [3] R. Olfati-Saber, J. Fax, and R. Murray, “Consensus and cooperation in networked multi-agent systems,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.
- [4] W. Ren, R. W. Beard, and D. B. Kingston, “Multi-agent Kalman consensus with relative uncertainty,” in *IEEE American Controls Conference*, vol. 3, 2005, pp. 1865–1870.
- [5] M. Alighanbari and J. P. How, “Unbiased Kalman consensus algorithm,” *Journal of Aerospace Computing Information and Control*, vol. 5, no. 9, pp. 209–311, 2008.
- [6] R. Olfati-Saber, E. Franco, E. Frazzoli, and J. Shamma, “Belief consensus and distributed hypothesis testing in sensor networks,” in *Network Embedded Sensing and Control*, vol. 331. Springer-Verlag, 2006, pp. 169–182.
- [7] L. F. Bertuccelli, “Robust decision making with model uncertainty in aerospace systems,” Ph.D. dissertation, Massachusetts Institute of Technology, 2008.
- [8] N. A. Lynch, *Distributed Algorithms*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1996.
- [9] M. DeGroot, “Reaching a consensus,” in *Journal of the American Statistical Association*, vol. 69, no. 345, 1974, pp. 118–121.

- [10] R. Winkler, “The consensus of subjective probability distributions,” in *Management Science*, vol. 15, no. 2, 1968, pp. 61–75.
- [11] —, “Combining probability distributions from dependent information sources,” in *Management Science*, vol. 27, no. 4, 1981, pp. 479–488.
- [12] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Transactions of the ASME—Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.
- [13] R. J. Aumann, “Agreeing to disagree,” in *The Annals of Statistics*, vol. 4, no. 6, 1976, pp. 1236–1239.
- [14] D. Castañón and D. Teneketzis, “Further results on the consensus problem,” in *IEEE Conference on Decision and Control*, vol. 26, Dec. 1987, pp. 1820–1825.
- [15] —, “Further results on the asymptotic agreement problem,” in *IEEE Transactions on Automatic Control*, vol. 33, no. 6, June 1988, pp. 515–523.
- [16] D. Teneketzis and P. Varaiya, “Consensus in distributed estimation with inconsistent beliefs,” in *Systems and Controls Letters*, vol. 4, 1984, pp. 217–221.
- [17] R. Washburn and D. Teneketzis, “Asymptotic agreement among communicating decisionmakers,” in *Stochastics*, vol. 13, 1984, pp. 103–129.
- [18] C. W. Reynolds, “Flocks, herds and schools: A distributed behavioral model,” in *SIGGRAPH '87: Proceedings of the 14th annual conference on computer graphics and interactive techniques*, 1987, pp. 25–34.
- [19] T. Vicsek, A. Czirook, E. Ben-Jacob, I. Cohen, and O. Shochet, “Novel type of phase transition in a system of self-driven particles,” *Phys. Rev. Lett.*, vol. 75, pp. 1226–1229, 1995.
- [20] 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.
- [21] R. Beard and V. Stepanyan, “Information consensus in distributed multiple vehicle coordinated control,” in *IEEE Conference on Decision and Control*, vol. 2, Dec. 2003, pp. 2029–2034.

- [22] J. Fax and R. Murray, “Information flow and cooperative control of vehicle formations,” *IEEE Transactions on Automatic Control*, vol. 49, no. 9, pp. 1465–1476, Sept. 2004.
- [23] R. Olfati-Saber and R. 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.
- [24] V. Blondel, J. Hendrickx, A. Olshevsky, and J. Tsitsiklis, “Convergence in multiagent coordination, consensus, and flocking,” in *IEEE Conference on Decision and Control*, Dec. 2005, pp. 2996–3000.
- [25] L. Moreau, “Stability of multiagent systems with time-dependent communication links,” *IEEE Transactions on Automatic Control*, vol. 50, no. 2, pp. 169–182, Feb. 2005.
- [26] W. Ren and R. Beard, “Consensus seeking in multiagent systems under dynamically changing interaction topologies,” *IEEE Transactions on Automatic Control*, vol. 50, no. 5, pp. 655–661, May 2005.
- [27] R. Olfati-Saber, “Ultrafast consensus in small-world networks,” in *IEEE American Controls Conference*, June 2005, pp. 2371–2378 vol. 4.
- [28] W. Ren, “Consensus based formation control strategies for multi-vehicle systems,” in *IEEE American Controls Conference*, June 2006, pp. 6–12.
- [29] Y. Cao, W. Ren, N. Sorensen, L. Ballard, A. Reiter, and J. Kennedy, “Experiments in consensus-based distributed cooperative control of multiple mobile robots,” in *International Conference on Mechatronics and Automation*, Aug. 2007, pp. 2819–2824.
- [30] P. Yang, R. Freeman, and K. Lynch, “Distributed cooperative active sensing using consensus filters,” in *IEEE International Conference on Robotics and Automation*, April 2007, pp. 405–410.
- [31] R. Freeman, P. Yang, and K. Lynch, “Stability and convergence properties of dynamic average consensus estimators,” in *IEEE Conference on Decision and Control*, Dec. 2006, pp. 338–343.
- [32] M. Zhu and S. Martinez, “Dynamic average consensus on synchronous communication networks,” in *IEEE American Controls Conference*, June 2008, pp. 4382–4387.

- [33] B. Rao and H. Durrant-Whyte, “A decentralized Bayesian algorithm for identification of tracked targets,” *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 23, no. 6, pp. 1683–1698, Nov/Dec 1993.
- [34] H. Durrant-Whyte, B. Rao, and H. Hu, “Toward a fully decentralized architecture for multi-sensor data fusion,” in *IEE Colloquium on Principles and Applications of Data Fusion*, Feb 1991, pp. 1–4.
- [35] S. Grime, H. Durrant-Whyte, and P. Ho, “Communication in decentralized data-fusion systems,” in *IEEE American Controls Conference*, vol. 4, June 1992.
- [36] S. Grime and H. Durrant-Whyte, “Data fusion in decentralized sensor networks,” *Control Engineering Practice*, vol. 2, no. 5, pp. 849 – 863, 1994.
- [37] A. Makarenko and H. Durrant-whyte, “Decentralized data fusion and control in active sensor networks,” in *International Conference on Information Fusion*, 2004.
- [38] A. Makarenko and H. Durrant-Whyte, “Decentralized Bayesian algorithms for active sensor networks,” *International Conference on Information Fusion*, vol. 7, no. 4, pp. 418 – 433, 2006.
- [39] L. Xiao, S. Boyd, and S. Lall, “A scheme for robust distributed sensor fusion based on average consensus,” in *International Symposium on Information Processing in Sensor Networks*, April 2005, pp. 63–70.
- [40] R. Olfati-saber, “Distributed Kalman filtering and sensor fusion in sensor networks,” in *Network Embedded Sensing and Control*, vol. 331. Springer-Verlag, 2006, pp. 157–167.
- [41] H. B. Mitchell, *Multi-Sensor Data Fusion: An Introduction*. Heidelberg, Germany: Springer, 2007.
- [42] M. E. Liggins, D. L. Hall, and J. Llinas, Eds., *Multisensor Data Fusion: Theory and Practice, 2nd Ed.* CRC Press, 2009.
- [43] R. Olfati-Saber and J. Shamma, “Consensus filters for sensor networks and distributed sensor fusion,” in *IEEE Conference on Decision and Control and European Control Conference*, Dec. 2005, pp. 6698–6703.

- [44] J. Tsitsiklis, D. Bertsekas, and M. Athans, “Distributed asynchronous deterministic and stochastic gradient optimization algorithms,” *IEEE Transactions on Automatic Control*, vol. 31, no. 9, pp. 803–812, Sep 1986.
- [45] R. A. Horn and C. R. Johnson, *Matrix Analysis*. Cambridge University Press, 1985.
- [46] A. L. Barker, D. E. Brown, and W. N. Martin, “Bayesian estimation and the Kalman filter,” *Computers and Mathematics with Applications*, vol. 30, pp. 55–77, 1994.
- [47] A. Gelman, J. Carlin, H. Stern, and D. Rubin, *Bayesian Data Analysis*, 2nd ed. Chapman and Hall, 2004.
- [48] R. Kass and L. Wasserman, “Formal rules for selecting prior distributions: A review and annotated bibliography,” *Journal of the American Statistical Association*, 1994.
- [49] C. P. Robert, *The Bayesian Choice: from decision-theoretic foundations to computational implementation*, 2nd ed. New York, NY, USA: Springer, 2001.
- [50] R. Dearden, N. Friedman, and D. Andre, “Model based Bayesian exploration,” in *Conference on Uncertainty in Artificial Intelligence*, 1999, pp. 150–159.
- [51] M. Strens, “A Bayesian framework for reinforcement learning,” in *International Conference on Machine Learning*, 2000, pp. 943–950.
- [52] C. S. R. Fraser, L. F. Bertuccelli, and J. P. How, “Reaching consensus with imprecise probabilities over a network,” in *AIAA Conference on Guidance, Navigation and Control*, Aug. 2009 (to appear).
- [53] N. Friedman and Y. Singer, “Efficient Bayesian parameter estimation in large discrete domains,” in *Conference on Advances in Neural Information Processing Systems II*, 1999, pp. 417–423.
- [54] L. F. Bertuccelli and J. P. How, “Reaching consensus with imprecise probabilities over a network,” MIT, Tech. Rep., 2008, <http://hdl.handle.net/1721.1/43949>.
- [55] R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.
- [56] C. J. Watkins, “Models of delayed reinforcement learning,” Ph.D. dissertation, Cambridge Univ., 1989.

- [57] R. Dearden, N. Friedman, and S. Russell, “Bayesian Q-learning,” in *In AAAI/IAAI*, 1998, pp. 761–768.
- [58] R. S. Sutton, “Integrated architectures for learning, planning, and reacting based on approximating dynamic programming,” in *Conference on Machine Learning*, 1990, pp. 216–224.
- [59] M. Valenti, B. Bethke, G. Fiore, J. How, and E. Feron, “Indoor multi-vehicle flight testbed for fault detection, isolation, and recovery,” in *AIAA Conference on Guidance, Navigation and Control*, August 2006.
- [60] J. How, B. Bethke, A. Frank, D. Dale, and J. Vian, “Real-time indoor autonomous vehicle test environment,” *IEEE Control System Magazine*, vol. 28, no. 2, pp. 51–64, April 2008.
- [61] J. How, C. Fraser, K. Kulling, L. Bertuccelli, O. Toupet, L. Brunet, A. Bachrach, and N. Roy, “Increasing autonomy of UAVs,” *IEEE Robotics & Automation Magazine*, vol. 16, no. 2, pp. 43–51, June 2009.
- [62] D. Dionne and C. A. Rabbath, “Multi-UAV decentralized task allocation with intermittent communications: the DTC algorithm,” in *IEEE American Controls Conference*, 2007.
- [63] M. Flint, T. Khovanova, and M. Curry, “Decentralized control using global optimization,” in *Proceedings of the AIAA*, 2007.
- [64] A. Ryan, J. Tisdale, M. Godwin, D. Coatta, D. Nguyen, S. Spry, R. Sengupta, and J. K. Hedrick, “Decentralized control of unmanned aerial vehicle collaborative sensing missions,” in *IEEE American Controls Conference*, 2007.
- [65] B. Grocholsky, J. Kellar, V. Kumar, and G. Pappas, “Cooperative air and ground surveillance,” *IEEE Robotics & Automation Magazine*, vol. 13, no. 3, 2006.
- [66] W. Wong, F. Bourgault, and T. Furukawa., “Multi-vehicle Bayesian search for multiple lost targets,” in *IEEE International Conference on Robotics and Automation*, 2005.
- [67] Y. Yang, A. Minai, and M. Polycarpou, “Evidential map-building approaches for multi-UAV cooperative search,” in *IEEE American Controls Conference*, 2005.

- [68] T. Furukawa, F. Bourgault, B. Lavis, and H. Durrant-Whyte, “Recursive Bayesian search-and-tracking using coordinated UAVs for lost targets,” in *IEEE International Conference on Robotics and Automation*, May 2006, pp. 2521–2526.
- [69] J. Elston and E. Frew, “Hierarchical distributed control for search and tracking by heterogeneous aerial robot networks,” in *IEEE International Conference on Robotics and Automation*, 2008, pp. 170–175.
- [70] F. Bourgault, T. Furukawa, and H. F. Durrant-Whyte., “Process model, constraints, and the coordinated search strategy.” *IEEE International Conference on Robotics and Automation*, 2004.
- [71] ———, “Decentralized Bayesian negotiation for cooperative search.” *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.
- [72] S. Brown, “Optimal search for a moving target in discrete time and space.” *Operations Research*, vol. 28, no. 6, 1980.
- [73] L. Stone, *Theory of Optimal Search*. Military Applications Society, 2004.
- [74] Y. B. Shalom, X. R. Li, and T. Kirubarajan., *Estimation with Applications to Tracking and Navigation*. Wiley Interscience, 2001.
- [75] T. Kirubarajan, Y. Bar-Shalom, K. R. Pattipati, I. Kadar, B. Abrams, and E. Eadan., “Tracking ground targets with road constraints using an IMM estimator.” *IEEE Aerospace Conference*, 1998.
- [76] L. Brunet, H. L. Choi, J. P. How, and M. Alighanbari, “Consensus-based auction approaches for decentralized task assignment,” *AIAA Conference on Guidance, Navigation and Control*, 2008.
- [77] H.-L. Choi, L. Brunet, and J. P. How, “Consensus-based decentralized auctions for robust task allocation,” *IEEE Transactions on Robotics*, 2009 (to appear).
- [78] Intel Corporation, “OpenCV computer vision library,” Available at <http://www.intel.com/technology/computing/opencv/>, 2007.
- [79] S. Arulampalam, N. Gordon, and B. Ristic, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Boston, MA: Artech House, 2004.

- [80] S. Park, J. Deyst, and J. P. How, “Performance and Lyapunov stability of a nonlinear path following guidance method,” *AIAA Journal of Guidance, Control, and Dynamics*, vol. 30, no. 6, pp. 1718–1728, Nov. 2007.
- [81] G. Ducard, K. Kulling, and H. P. Geering, “A simple and adaptive on-line path planning system for a UAV,” in *Mediterranean Conference on Control & Automation*, 2007.
- [82] Vicon, “Vicon MX systems,” Available at <http://www.vicon.com/products/viconmx.html>, July 2006.
- [83] Motion Analysis, “Motion analysis raptor 4 digital real-time system,” Available at <http://www.motionanalysis.com/pdf/specs/raptor4.pdf>.
- [84] Ascending Technologies, “Asctec hummingbird quadcopter,” Available at <http://www.asctec.de>, 2008.
- [85] R. Mehra, “Approaches to adaptive filtering,” *IEEE Transactions on Automatic Control*, vol. 17, no. 5, pp. 693–698, Oct 1972.
- [86] P. S. Maybeck, *Stochastic Models, Estimation and Control*. Academic Press, 1982, vol. 2.
- [87] P. J. Huber, *Robust Statistics*. Wiley, 1981.
- [88] Y. Yang, M. Polycarpou, and A. A. Minai., “Decentralized cooperative search in uav’s using opportunistic learning,” *AIAA Conference on Guidance, Navigation and Control*, 2002.
- [89] Y. Yang, A. A. Minai, and M. Polycarpou, “Decentralized cooperative search by networked UAVs in an uncertain environment,” *IEEE American Controls Conference*, 2004.
- [90] L. Bertuccelli and J. How, “Robust UAV search for environments with imprecise probability maps,” in *IEEE Conference on Decision and Control*, Dec. 2005, pp. 5680–5685.
- [91] T. Yanagimoto and T. Ohnishi, “Extensions of the conjugate prior through the Kullback-Leibler separators,” *Journal of Multivariate Analysis*, vol. 92, no. 1, pp. 116 – 133, 2005.



- [92] V. Saligrama and D. Castañón, “Reliable distributed estimation with intermittent communications,” in *IEEE Conference on Decision and Control*, Dec. 2006, pp. 6763–6768.
- [93] R. Rahman, M. Alanyali, and V. Saligrama, “Distributed tracking in multihop sensor networks with communication delays,” *IEEE Transactions on Signal Processing*, vol. 55, no. 9, pp. 4656–4668, Sept. 2007.
- [94] D. J. Salmond, “Tracking in uncertain environments,” Royal Aircraft Establishment, Farnborough, UK, Tech. Rep., 1989.