COMMUNICATION EFFICIENT DECENTRALIZED INFORMATION FUSION IN SENSOR NETWORKS

By

RAKSHIT D. ALLAMRAJU

Bachelors of Technology in Avionics Engineering

University of Petroleum and Energy Studies

Dehradun, India

2012

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
December, 2015

COMMUNICATION EFFICIENT DECENTRALIZED INFORMATION FUSION IN SENSOR NETWORKS

Thesis Approved:

Dr.Girish Chowdhary

Thesis Advisor

Dr.Christopher Crick

Dr.He Bai

Name: RAKSHIT D. ALLAMRAJU

Date of Degree: DECEMBER, 2015

Title of Study: COMMUNICATION EFFICIENT DECENTRALIZED INFORMATION FUSION IN SENSOR NETWORKS

Major Field: MECHANICAL AND AEROSPACE ENGINEERING

Abstract: Many physical, geological and environmental phenomena are spread over large spatio-temporal scales and require robust sensor network systems to gather data, collaborate and monitor the area for anomaly detection. In decentralized sensor networks, deployed in monitoring such events, agents are constrained to monitor localized regions and collaborate with other agents to obtain a common global model. While such sensor networks yield significant benefits such as endurance and scalability, they are constrained in amount of on-board resources available such as computational ability, communication bandwidth and fuel. In this work the problem of decentralized functional inference over a spatially separated sensor network with limited communication capability is studied. Gaussian Processes (GPs) have been studied as priors over spatially distributed functions. The key benefit of GPs is that the number and location of regression kernels, and the numerical values of associated weights, are simultaneously inferred from the underlying data. While this enables the model to adapt its structure based on the data, it also makes decentralized inference using Consensus type algorithms difficult if agents do not know each-other's kernel selection a-priori, and traditional sample based inference is communication-inefficient if all of the data is shared between agents. A new decentralized communication-efficient algorithm for decentralized information fusion over GPs is presented of which the key contribution is that agents do not have to a-priori know each-other's kernel selections. Instead, our algorithm enables agent's to build a global GP model by fusing together local compressed GPs. To further prevent unnecessary communication, the presented algorithm can utilize information theoretic measures on value-of-information to initiate broadcasts only when agents have sufficient new information. The algorithm is compared with several state of the art methods on real-world and synthetic datasets, and is shown to lead to efficient estimation accuracy with decreased communication cost, without having to assume that agents share a common set of kernels. To further evaluate the real time performance of the presented method, a multi-agent simulator system is developed as a test bench. Experiments for decentralized multi-agent planning, which use the described fusion algorithm, are conducted and the respective results are provided.

Acknowledgments

I would like to thank my parents, Dayal Das and Jaya Lakshmi Allamraju, for their constant support and encouragement in letting me pursue my Masters. Without their full support, I would have never realized my dream of obtaining an advanced degree.

I would like to thank my sister and brother-in-law, Sushama and Dileep Tata, for their motivation and support throughout my entire duration of my Masters and constantly boosting my morale when things didn't look good. I wouldn't have got this work completed without their daily pep talk.

My advisor, Dr.Girish Chowdhary, is among some of the most erudite persons I have met. His infinite patience when teaching me how to properly identify, analyze and approach the problem with an out-of-the-box thinking approach, is something I am really grateful for. Dr.Girish taught me a lot on how to research properly, by always providing the best of suggestions to ensure I not simply do research, but also present them well.

I would like to thank my committee member Dr.Christopher Crick for his support in various collaboration projects and helping me out whenever we hit roadblocks in our work. His methods of explaining the idea were always clear and amazing. I would also like to thank my other committee member Dr.He Bai for his inputs in my research, results and my thesis document.

My friends in the lab, Ben Reish, Ali Abdollahi, Sri Vuppala, Harshal Maske, Alex Suhren, Allan Axelrod, Talpasai, Milecia Matthews and Dane Johnson. They provided all help they could from their areas of expertise, which helped me put this thesis together. I had some of the best times discussing various ideas with you guys, which definitely got me excited with new concepts and disciplines to explore.

And of course my friends outside the lab, Shyam, Kavya, Emilio, Orlando, Rachana, Sravani and Lalitha, who always put up with my eccentric behavior and made sure I maintained my sanity. I will cherish all the good times we had together.

This work was supported in part by AFOSR grant FA9550-15-1-0146 and DOE grant DE-FE0012173

Contents

List of Figures

Chapter 1

Introduction

This thesis examines the problem of decentralized information fusion in spatially separated sensor networks with communication constraints. A novel Gaussian Process based model fusion method is proposed which presents a solution to the problem decentralized estimation by fusing two Gaussian Processes and reducing the communication cost incurred in the network.

1.1 Motivation

The rapid growth of cheap sensing devices has fueled a trend to use Autonomous Sensor Networks (ASN) to perform complex missions. For example, in applications such as carbon sequestration, sensors are required to monitor the variations in carbon dioxide levels over underground CO_2 storage locations and learn the spatio-temporal variations in CO_2 over a large area, so that CO_2 leaks in the storage units can be quickly detected. Physics based models alone often do not have sufficient fidelity to model and predict such large phenomena, necessitating the need to utilize data-driven modeling and inference. However, the massive scales of such environmental phenomena and the Big Data they generate pose key barriers in learning and predicting spatio-temporal variations. The use of a team of spatially distributed agents, each of which can estimate a small section of the function, and autonomously collaborate to estimate the global phenomena is desired.

In the sensing literature, authors have argued that a decentralized network can improve coverage, robustness, and be more adaptive to changes, if it operates in a decentralized manner, over a single sensor. Decentralized sensor network systems posses several capabilities, such as lack of a-priori knowledge of global network topology allowing nodes to be constructed in a modular fashion, flexibility to on-line addition or loss of sensors and scalability to large sensor nodes [10, 16] which makes them more attractive in comparison to centralized sensor networks, particular in application such as monitoring and surveillance. Figure 1.1 shows a decentralized sensor network with agents estimating local models of an unknown function and passing them to their neighbors.

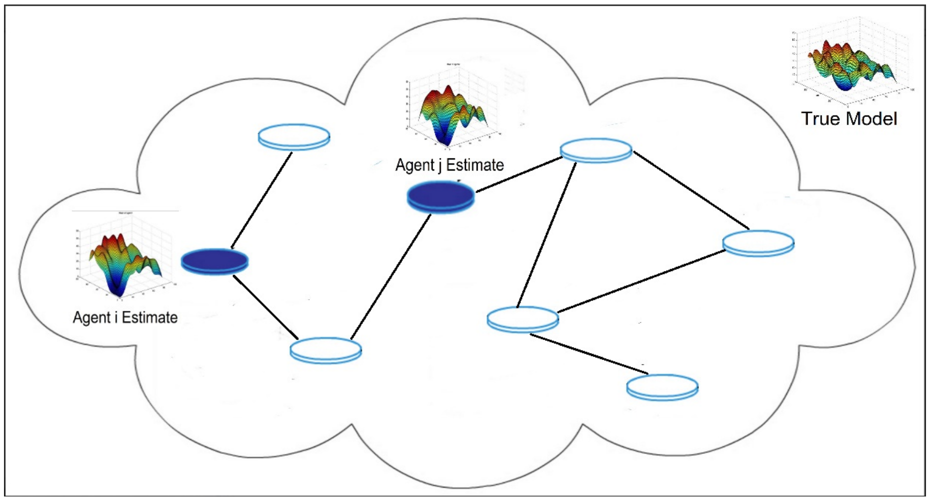


Figure 1.1: Sensor Network for decentralized functional inference in spatially separated sensors. Agents communicate local function estimates f_i to estimate the global function f.

A key benefit of a decentralized network is that agents can adapt to their local surroundings. For example, when learning local CO_2 distributions using a Gaussian Process [45], agents are free to choose the locations of their own kernels to better adapt the model to their environment. The main challenge in this situation is in ensuring that agents can fuse together the local models they have learned into a common coherent global model. A naive way to achieve this would be to transmit all of the data that the agents have used to build their model to neighboring agents. However, this is communication intensive, and not suitable for resource-constrained platforms, since transmission of individual data samples consumes significant amount of power. Our work is interested in techniques which can monitor and detect spatial phenomena over large periods of time with minimum communication costs.

Designing sensor network for spatial estimation tasks has been previously studied in environmental monitoring applications. In Geo-statistics, spatial processes modeled as random fields are estimated via simple Kriging interpolation techniques [14, 49],alternatively known as Gaussian Process Bayesian Non Parametric(GP-BNP) models. However, with decentralized function estimation using simple Kriging techniques presented in literature, communication between agents is assumed to be continuous and hence these techniques are not very efficient when network is communication restricted.

In this thesis a solution is presented to the problem of decentralized functional inference using

Gaussian Process(GP) priors over a spatially separated sensor network. Gaussian Processes(GPs) are Bayesian Nonparametric (BNP) models that have been widely studied as priors for functional inference [2, 13, 45] . They lead to data-driven models in which functions are modeled using covariance kernel between any two data points to build generative models of smooth spatially correlated distributions. The key benefit of GPs, and other kernel based models [28, 48], is that the number and location of regression kernels, and the numerical values of the associated regression weights are simultaneously inferred from the data. This enables the model to adapt its structure to best reflect the data, without having to assume a lot of a-priori domain knowledge. In the sense that the number of parameters of the model are not fixed a-priori, rather adjusts to the data, GPs are termed as nonparametric models.

However, while this property of GPs makes them very powerful and flexible regression tools, it also makes the problem of decentralized inference over GPs very hard. The main reason behind this is that since each agent sees a different set of data, the kernels it chooses are likely to be different than other agents, and it is not clear as to how the correlation between kernels should be accounted for when fusing together two different GP models. In particular, Consensus type algorithms are difficult to implement without having to assume a common set of kernels [8, 11].

Another issue which arises in deploying a team of decentralized autonomous sensor networks for spatial modeling include coordination between sensors to optimally gather information, cooperation to obtain unknown information and respond to unanticipated situations or changes in the environment that are sensed. As the environment changes, the agents on the team must be in agreement as to what changes took place. A consequence of the above challenges is for the network to maintain information sharing between agents to perform desired task in an optimal manner. However, when communication is limited, agents will have incomplete information about the underlying function and each agent is not guaranteed to have the same estimate of the function which may lead to suboptimal coordination between agents in the ASN.

1.2 Background

This section provides a background on Gaussian Process inference on function f and lays out the problem formulation for the decentralized information fusion in autonomous sensor networks.

1.2.1 Preliminaries: Gaussian Process Regression

Gaussian Processes(GPs) are Bayesian Nonparametric models that have proven to be powerful and flexible priors for functional regression [45,48]. A GP is defined as a collection of random variables such that every finite subset is jointly Gaussian. The joint Gaussian condition means that GPs are completely characterized by their second order statistics [45]. A GP is a distribution over functions, that is, a draw from a GP is a function. For the sake of clarity of exposition, we will assume that $f \in \Re$; the extension to the multidimensional case are available [3,25].

$$f(\cdot) \sim \mathscr{GP}(m(\cdot), k(\cdot,\cdot)), \tag{1.1}$$

where $m(\cdot)$ is the mean function, and $k(\cdot,\cdot)$ is a real-valued, positive definite covariance kernel function. Under GP regression, the mean is assumed to lie in the class of functions

$$\mathscr{G} = \left\{ g(\cdot) \in \Re^{\mathscr{X}} \;\middle|\; g(\cdot) = \sum_{i=1}^{\infty} \alpha_i k(z_i, \cdot) \right\}, \tag{1.2}$$

where $\mathscr{X} = \Re^n$, $\alpha_i \in \Re$, $z_i \in \mathscr{X}$. The space $\mathscr{G}$ is a subspace of $\mathscr{H}$, an RKHS, and $\|g\|_{\mathscr{H}} < \infty$ where $\|g(\cdot)\|^2_{\mathscr{H}} = \sum_{i=1}^{\infty}\sum_{j=1}^{\infty} \alpha_i \alpha_j k(z_i, z_j)$. This assumption imposes a smoothness prior on the mean and makes the problem amenable to analysis though the representer theorem [47].

Let $Z_\tau = \{z_1, \ldots, z_\tau\}$ be a set of state measurements, discretely sampled where $\{1 \ldots \tau\}$ are indices for the discrete sample times $\{t_1, \ldots, t_\tau\}$. The set defines a covariance matrix $K_{ij} := k(z_i, z_j)$. The positive definite function k generates a mapping ψ to an RKHS $\mathscr{H}$ such that $k(z_i, z_j) = \langle \psi(z_i), \psi(z_j) \rangle_{\mathscr{H}}$. GP regression assumes that the uncertainty in the data and the model follow Gaussian distributions, while modeling the function estimate using a mean function $\hat{m}$ and a covariance function $\hat{\Sigma}$. Since the observations are Gaussian, the likelihood function $p(y_\tau | Z_\tau, \beta)$ is Gaussian. The initial prior is set to $p(\beta) \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma_w})$, and Bayes' rule is used to infer the posterior distribution $p(\beta | Z_\tau, y_\tau)$ with each new observation. Since the posterior is Gaussian, the update generates a revised mean $\hat{m}_\tau$ and covariance $\hat{\Sigma}_\tau$. If $|Z_\tau|$ is finite, the solution for the posterior mean and covariance is also finite [47]. In particular, given a new input $z_{\tau+1}$, the joint distribution of the data available up to τ and z_τ under the prior distribution is

$$\begin{bmatrix} y_\tau \\ y_{\tau+1} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(Z_\tau, Z_\tau) + \omega^2 I & k_{z_{\tau+1}} \\ k^T_{z_{\tau+1}} & k^*_{\tau+1} \end{bmatrix}\right), \tag{1.3}$$

where $k_{z_{\tau+1}} = K(z_{\tau+1}, Z_\tau)$ and $k^*_{\tau+1} = k(z_{\tau+1}, z_{\tau+1})$. The posterior (sometimes called the predictive) distribution, obtained by conditioning the joint Gaussian prior distribution over the observation z_{t+1}, is computed by

$$p(y_{\tau+1}|Z_\tau, y_\tau, z_{\tau+1}) \sim \mathscr{N}(\hat{m}_{\tau+1}, \hat{\Sigma}_{\tau+1}), \tag{1.4}$$

where

$$\hat{m}_{\tau+1} = \beta^T_{\tau+1} k_{z_{\tau+1}} \tag{1.5}$$

$$\hat{\Sigma}_{\tau+1} = k^*_{\tau+1} - k^T_{z_{\tau+1}} C_\tau k_{z_{\tau+1}} \tag{1.6}$$

are the updated mean and covariance estimates, respectively, and where $C_\tau := (K(Z_\tau, Z_\tau) + \omega^2 I)^{-1}$ and $\beta_{\tau+1} := C_\tau y_\tau$.

Since both Z_τ and y_τ grow with data, computing the inverse becomes computationally intractable over time. Therefore, to adapt GPs for the online setting, we use the efficient and recursive scheme introduced in [15]. The set Z generates a family of functions $\mathscr{F}_Z \subset \mathscr{H}$ whose richness characterizes the quality of the posterior inference; therefore a natural and simple way to determine whether to add a new point to the subspace is to check how well it is approximated by the elements in Z. This is known as the kernel linear independence test [15], and is computed by

$$\gamma_{\tau+1} = \left\| \sum_{i=1}^{\tau} \alpha_i \psi(z_i) - \psi(z_{\tau+1}) \right\|^2_{\mathscr{H}}. \tag{1.7}$$

The scalar $\gamma_{\tau+1}$ is the length of the residual of $\psi(z_{\tau+1})$ projected onto the subspace $\mathscr{F}_{Z_\tau}$. When $\gamma_{\tau+1}$ is larger than a specified threshold, then a new data point should be added to the data set. The coefficient vector α minimizing (1.7) is given by $\alpha_\tau = K^{-1}_{Z_\tau} k_{z_{\tau+1}}$, meaning that

$$\gamma_{\tau+1} = k^*_{\tau+1} - k^T_{z_{\tau+1}} \alpha_\tau. \tag{1.8}$$

This restricted set of selected elements, called the *basis vector set*, is denoted by $\mathscr{BV}$. When incorporating a new data point into the GP model, the inverse kernel matrix can be recomputed with a rank-1 update.

In the next subsection the decentralized information fusion problem, for inference on an unknown function f, monitored by a sensor network, is presented.

1.2.2 Problem Formulation: Decentralized Information Fusion

Let f be an unknown function, defined on domain $\mathcal{X}$, monitored by N autonomous sensing agents. Each agent node $i \in \{1, 2, \ldots, N\}$, defines a local GP prior $p_i(f|\mathcal{X}) = \mathcal{N}(\mathbf{0}, \bar{\Sigma}_i)$, over the function and can perform a Bayesian update on the prior with a set of independent sensor observations $\bar{y}^i = \{y_1, y_2, \ldots, y_k\}$ with likelihood $\mathbb{P}(\bar{y}^i|\mathcal{X}_i, f)$; where $\mathcal{X}_i \subset \mathcal{X}$ are the sensing locations monitored by agent i. The posterior distribution on function f, by sensing agent i, over the domain $\mathcal{X}$ is given by

$$\mathbb{P}_i(f|\bar{y}^i, \mathcal{X}, \mathcal{X}_i) = \mathcal{N}(\mu_i, \Sigma_i) \tag{1.9}$$

where $\mu_i = k(\mathcal{X}, \mathcal{X}_i)((K(\mathcal{X}_i, \mathcal{X}_i)+\omega^2 I)^{-1})\bar{y}^i$ and $\Sigma_i = k(\mathcal{X}, \mathcal{X}) - k(\mathcal{X}, \mathcal{X}_i)((K(\mathcal{X}_i, \mathcal{X}_i)+\omega^2 I)^{-1})k(\mathcal{X}_i, \mathcal{X})$.

Given some agent-to-agent communication topology, assume that agent i is aware only of its connected neighbors and unaware of the entire communication topology. Let $N(i)$ represent the set the connected neighbors of agent i, and let Z_t^i represent the information set contained in i till time t i.e Z_t^i represents the local measurements $\bar{y}_t^i$ of agent i plus the information received by i from its neighbors till time $t-1$. The Gaussian Process decentralized information fusion problem is to find the posterior predictive distribution conditioned as

$$\mathbb{P}_{i,N(i)}(f) = \mathbb{P}\bigg(f|Z_t^i \bigcup_{j\in N(i)} Z_t^j,\, \mathcal{X}\bigg) \tag{1.10}$$

If all agents $\{j;\, \forall j \in N(i)\ \}$, communicate measurement samples $\bar{y}^j$, then computing the fused posterior distribution at agent i for time t would be straight forward since it involves a recursive Bayesian update of the local GP prior with the comprehensive set of observations at node i. Communicating observation samples between agents can however lead to huge communication costs. In order to overcome this problem, the decentralized information fusion problem (1.10) is reformulated such that the posterior distribution can be computed without the measurements samples. Assume agent i updates its GP with the information set from $j \in N(i); \forall j$ via a FIFO queue. The local GP estimate (1.9) at time t is redefined as follows

$$\mathbb{P}_i(f|Z_t^i) \propto \mathbb{P}(f|Z_t^i \cap Z_t^j)\mathbb{P}(f|Z_t^{i/j}) \tag{1.11}$$

The distribution $\mathbb{P}(f|Z_t^{i/j})$ is the posterior over f conditioned on information in i relative to j and $\mathbb{P}(f|Z_t^i \cap Z_t^j)$ is the posterior GP conditioned on the information common to both i and j. The joint fused posterior GP (1.10), between agent i and j, can be recovered using the factorization (1.11) as

$$\begin{aligned}\mathbb{P}_{i,j}(f) &= \mathbb{P}(f|Z_t^i \cup Z_t^j)\\ &\propto \mathbb{P}(f|Z_t^i \cap Z_t^j)\mathbb{P}(f|Z_t^{i/j})\mathbb{P}(f|Z_t^{j/i})\\ &= \frac{\mathbb{P}_i(f|Z_t^i)\mathbb{P}_j(f|Z_t^j)}{\mathbb{P}(f|Z_t^i \cap Z_t^j)}; \forall j \in N(i) \end{aligned} \tag{1.12}$$

The posterior distribution in equation (1.12) can be computed since the local GP distribution for each agent belong to a class of exponential functions i.e $\mathbb{P}_i(f|Z_t^i) := \mathcal{N}(\mu_{\mathbf{i}}, \mathbf{\Sigma_i})$ and $\mathbb{P}_j(f|Z_t^j) := \mathcal{N}(\mu_{\mathbf{j}}, \mathbf{\Sigma_j})$. However, to compute the joint common posterior $\mathbb{P}(f|Z_t^i \cap Z_t^j)$, the networks needs to communicate Z_t^i and Z_t^j, which still results in large communication costs.

Double counting of information is an important problem in Decentralized Data Fusion algorithms, since duplication of information results in a large communication penalty with little information propagation as messages which consist redundant information are exchanged between agents [1]. Methods such as Covariance intersection have been proposed which are shown to be robust to information duplication [12, 20]. These methods are however not suitable for functional estimation over large spaces as it is computationally intensive to calculate the inversion of the covariance matrix for the entire domain. Other methods such as Consensus [38, 46] and Distributed inference using graphical models [34,36] can be implemented with some modification to learn an unknown function at every location in $\mathcal{X}$. These algorithms however will result in high communication cost which are unsuitable when agents in sensor network are communication constrained. A method to reduce communication using Censoring is described in [34] for graphical model based inference. Mu et al. [36] have extended the censoring idea to a decentralized setting by adaptively adjusting a Value-of-Information (VoI) threshold based on the communication cost in the network. However the possible extensions of this method to perform function inference has never been put in practice.

1.2.3 Notations

The information flow between the set of agents in $I = \{1, 2, ..., N\}$ in a sensor network can be mathematically modeled using a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Set $\mathcal{V} = \{1, 2, \ldots, N\}$ denote vertex set and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ denotes the edge set. When agent i can communicate with agent j the vertex pair $(i, j) \in \mathcal{E}$. When $(i, j) \in \mathcal{E}$, agent j is called a neighbor of agent i and is denoted by $N(i)$. We assume that $\mathcal{G}$ is strongly connected i.e. there exist a path from i to j in each direction which can be formed from the elements in $\mathcal{E}$. The Gaussian process generated from information sampled by agent i is denoted as $\mathscr{GP}_i$ and the composite model obtained by fusing model of j into i is denoted as $\mathscr{G}\mathscr{P}_{ji};\ j \in N(i)$.

1.3 Related Work

1.3.1 Decentralized Data Fusion

Decentralized data fusion has received significant attention in the estimation and tracking literature [17,43] and many solutions have been proposed to solve the state estimation problem. We however, focus on the decentralized information fusion for inference on functions in our work. Current state of the art techniques for information fusion in decentralized sensor networks have been Channel Filters [30, 31], Covariance Intersection [22], Consensus [38, 46], and techniques for inference on Graphical models with censoring based approaches [27, 36]. These methods, however are mostly focused on inference for finite-dimensional states and need modifications to applied to function inference. Some recent works in modeling functions have been [11, 29]. These methods however assumes that each agent in the network have an a-priori fixed common model supports which is restrictive.

Channel Filters(CF) have been proposed as a solution to the problem of decentralized estimation and tracking with fusion of information between the nodes of the sensor network [30, 41]. In CF algorithms an unknown state x, such as target position, is estimated with nodes estimating part of the true state x. Each sensor node receives information from its neighbors and assimilates this information before passing it to subsequent nodes. Thus even as the node's neighbor list grows,the size of the outgoing message remains constant. This allows the network to scale indefinitely. In many information fusion methods, a key challenge is to prevent double counting of information [41]. The CF system maintains a synchronization between the local information at node and the previously communicated information to the sensor. The main power of CF algorithms is that they eliminate double counting of information by adhering to a tree structure. However maintaining coordination on the beliefs leads to huge communication costs due to continuous information exchange.

A second set of algorithms which prevents double counting of information is Covariance Intersection (CI). These methods achieve fusion between any two estimates when the correlation between them is unknown. These methods have been used in SLAM applications [23] and remote sensing for image fusion [22]. CI filters are designed to work with low dimensional Gaussian distributions and they do not scale well for high dimensional distributions.

Decentralized information fusion to model demand in mobility-on-demand systems, using a fleet of autonomous robotic vehicles, has been studied in [11]. In this work a each agent receives demand measurements from different sectors and models these measurements with a Gaussian Process. Each agent is assumed to have a common set of kernels for the GP model. Agents then communicate local GP summaries, which include information about the model to their neighbors and include neighbor's summaries into their models to facilitate greater predictive capability of demand in unobserved locations. Our work is closely related to the information fusion methodology for inference described in [11] in the sense that we also focus on fusing compressed GP models instead of transmitting the data itself. The crucial difference however is that we relax the assumptions of common support for the GP model and one-one communication topology. This ability makes our algorithm truly decentralized, as agents do not need to agree on common kernel set a-priori, rather, agents are free to adapt the model structure to their individual environments.

1.3.2 Communication Efficient Distributed Inference

In Graphical models, agents build local probability models on the parameters of the state. When new measurements are observed, agents propagate messages about the measurements between each other to update their probability beliefs utilizing a priori known information about the correlations between each other's probability models. Noting that not all agents have valuable information to communicate at all times, the idea of censoring (i.e. stopping from communicating) low information-value agents has been used in distributed inference algorithms [34–36, 51]. In censoring based methods, agents use information-theoretic metrics on Value-of-Information (VoI) and agents are censored from broadcasting their information if they have low Value-of-Information. For example, in the recently published A-VoIDS algorithms of Mu et al. agents use KL-divergence between prior and posterior estimates to self-censor low value measurements [36]. The censoring threshold is adapted to available communication bandwidth to ensure fastest possible convergence in communication restricted environments and AVoIDS was demonstrated to outperform consensus and random broadcast algorithms for problems involving the estimation of hyperparameters of a commonly observed probability distribution over a scalar quantity. Theoretically distributed inference using graphical methods, like AVoIDs, can be extended to functional inference. However, a naive extension may not robust to rumor propagation i.e, propagation of incorrect estimates to all agents in the network, and double counting information.

1.4 Contributions

The contributions of this thesis are as follows. In Chapter 3, a novel algorithm for decentralized GP inference, which is termed as GP-Fusion (GPF), and is designed to operate under strict communication constraints. GP-Fusion is based on the idea that instead of transmitting the data and incurring a large communication penalty, agents directly combine local models of other agents into their own models, which reduces communication cost significantly. Theoretical guarantees on the error in function estimated at local node and the centralized node show that this error is upper bounded. In Chapter 4 it is demonstrated empirically, on one synthetic and two real world datasets, that GP-Fusion has orders of magnitude lower communication cost while still possessing a good accuracy in estimating the function.

In Chapter 5 a high fidelity simulation framework called *MAGE: Multi-Agent Game Emulator* developed at Distributed Autonomous Systems Lab(DAS-Lab), in Oklahoma State University, is proposed. MAGE brings together FlightGear's JSBSim flight-physics simulation engine, Simgear 3D visualization engine, and the Qground Ground Control Station interface to create a unique high-fidelity multi-agent flight simulation environment that enables multiple manned and unmanned aerial agents to "practice" together over a wide array of mission scenarios. The effectiveness of GP-Fusion in real mobile sensing platforms is tested on MAGE using an example mission scenario: the spatial estimation of an unknown risk function due to presence of hostile entities in some regions. A simple planning algorithm, which uses the output of model inference demonstrates the effectiveness of the method.

Chapter 2

Decentralized functional estimation algorithms

In order to evaluate the decentralized information fusion algorithm presented in this paper, a set of baseline algorithms are presented for comparison in this section.

2.1 Weighted Average Consensus

The first method proposed is a consensus-based framework for estimating the unknown function f, at each node, in a decentralized manner. This framework implements a Generalized Kalman Consensus Filter (GKCF) algorithm [24], which is a state-of-the-art distributed algorithm for estimating an unknown state by fusing multiple measurements from different sensors. GKCF is an extension of the Kalman Consensus Filter algorithm to improve performance of estimation in presence of naive nodes and duplication of information. In GKCF, each sensor's mean estimate is weighted based on the information at that node and hence agents will update their parameters toward high information nodes.

In a decentralized estimation using GKCF, the nodes are assumed to have peer-to-peer communication channels. Thus, when a sensor i gets new measurements from the underlying process, say $\bar{y}_i$, it shares this measurement information with its network neighbors $N(i)$. This measurement information is used to update the estimate model parameters at each node receives measurement of $\bar{y}_i$. By defining the function f as the unknown state, which a set of agents are monitoring, the GKCF framework can be used to estimate the f at each node, in a decentralized manner.

In implementing the GKCF, the network is assumed to monitor a function whose state dynamics are modeled as

$$f[t+1] = \Phi f[t] + \gamma[t] \tag{2.1}$$

where Φ is the state transition matrix and the process noise $\gamma[t]$ is modeled as $\mathcal{N}(\mathbf{0}, \mathbf{Q})$; Each agent obtains measurements from the function f through observation model

$$y_i = \mathbf{H}_i f + \nu_i \tag{2.2}$$

For ease of exposition, the function is considered as time invariant (i.e. f is stationary) and hence the state transition matrix is set to be unity. The observation matrix $\mathbf{H}_i$ is defined to be one at locations where agents can obtain measurements from f and the noise ν_i is modeled as a zero mean Gaussian noise with covariance $\mathbf{R}_i$.

In Weighted Average Consensus each agent starts out with a prior estimate μ_i^- and its covariance $\mathbf{\Sigma}_i^-$. The inverse of the covariance $\mathbf{\Sigma}_i$ is also defined as the information matrices as they measure the confidence of the agents based on their measurements. At each time step, agent i obtains measurements from the function and computes an information vector and matrices $\mathbf{u}_i$ and $\mathbf{U}_i$ and broadcasts these parameters, along with its current estimate μ_i^- and covariance $\mathbf{\Sigma}_i^-$ to its neighbors. By defining the weighted state and weight for each agent as $\mathbf{v}_i = \mathbf{\Sigma}_i^- \mu_i^-$ and $\mathbf{V}_i = \mathbf{\Sigma}_i^-$, the Weighted Average Consensus algorithm states that performing updates

$$\mathbf{v}_i[k] = \mathbf{v}_i[k-1] + \epsilon \sum_{j \in N(i)} (\mathbf{v}_j[k-1] - \mathbf{v}_i[k-1]) \tag{2.3}$$

$$\mathbf{V}_i[k] = \mathbf{V}_i[k-1] + \epsilon \sum_{j \in N(i)} (\mathbf{V}_j[k-1] - \mathbf{V}_i[k-1]) \tag{2.4}$$

Using (2.3) and (2.4) it can be shown that for all agents $i \in \mathcal{V}$, the estimate $\mathbf{\Sigma}_i^{-1}[k]\mu_i[k]$ converge as $k \to \infty$ [8, 33, 37]. The term ϵ is a rate parameter and should lie between 0 and $\frac{1}{\Delta_{max}}$; where Δ_{max} is the maximum degree of the network $\mathcal{G}$. In a decentralized setting, it is sufficient that agents communicate the weighted state and information matrices to their neighbors alone. The Weighted Average Consensus algorithm is shown in algorithm 1.

Algorithm 1 Weighted Average Consensus

```
1: procedure DEFINE ITERATIONS $k$, RATE PARAMETER $\epsilon$, INITIAL COVARIANCE $\mathbf{\Sigma}_i^-(0)$, INITIAL ESTIMATE $\mu_i^-(0)$
2:     for each time $t = 1, 2, \ldots, T$ do
3:         for each agent $i \in \mathcal{V}$ do
4:             Get measurements $y_i$
5:             Compute information vector and matrices as
6:             $\mathbf{u}_i = \mathbf{H}_i^T \mathbf{R}_i^{-1} y_i$
7:             $\mathbf{U}_i = \mathbf{H}_i^T \mathbf{R}_i^{-1} \mathbf{H}_i$
8:             Broadcast $\mathbf{u}_i$ and $\mathbf{U}_i$ and receive $\mathbf{u}_j$ and $\mathbf{U}_j$ from neighbors $j \in N(i)$
9:             For each neighbor receiving a local information from $i$, increment communication cost
10:            $C[t] = C[t] + \bar{c}_i; \forall j \in N(i)$
11:            Fuse the information vector and matrices with local information
12:            $\mathbf{b}_i = \sum_{j' \in N(i) \cup i} \mathbf{u}'_j$
13:            $\mathbf{B}_i = \sum_{j' \in N(i) \cup i} \mathbf{U}'_j$
14:            Initialize consensus variables $\mathbf{v}_i[0]$ and $\mathbf{V}_i[0]$ as
15:            $\mathbf{v}_i[0] = \mathbf{\Sigma}_i^- \mu_i^-$
16:            $\mathbf{V}_i[0] = \mathbf{\Sigma}_i^-$
17:            For $k$ iterations run weighted consensus on $\mathbf{V}_i[k]$ and $\mathbf{v}_i[k]$ using equations (2.3) and (2.4) and update the prior state and covariance
18:            $\mu_i^- = \mathbf{V}_i[k]^{-1} \mathbf{v}_i[k]$
19:            $\mathbf{\Sigma}_i = \mathbf{V}_i[k]$
20:            Compute the GKCF estimates
21:            $\mathbf{\Sigma}_i^+ = \mathbf{\Sigma}_i^- + B_i$
22:            $\mu_i^+ = \mu^- + \mathbf{\Sigma}_i^{+^{-1}}(\mathbf{b}_i - \mathbf{B}_i \mu_i)$
23:            Propagate estimates to next time step
24:            $\mathbf{\Sigma}_i^-(t+1) = (\Phi^T \mathbf{\Sigma}_i^+(t) \Phi + Q)^{-1}$
25:            $\mu_i^-(t+1) = \Phi \mu_i^+(t)$
26:        end for
27:    end for
28: end procedure
```

2.2 Adaptive Value of Information based Distributed Sensing (AVoIDS)

The second algorithm presented for baseline comparison is a multidimensional extension of the Adaptive Value of Information based Distributed Sensing(AVoIDs) algorithm presented in Mu et.al. [36]. This algorithm is a state-of-the-art method for estimating an unknown state of interest by minimizing the amount of communication required in the decentralized sensor network. In AVoIDs, each agent starts off with the same global prior $\mathbb{P}(f|\mu, \Sigma)$ on f. At each time t, agents obtain a set of measurements which are stored in a local buffer $\mathbf{S}_i[t]$ instead of relaying to neighbors immediately. To broadcast, a measure on the value of the measurements, termed *Value of Information(VoI)*, is calculated between the prior and the posterior based on the buffered measurements i.e. $\mathbb{P}(f|\mu, \Sigma, \mathbf{S}_i[t])$. A metric to calculate the VoI($V_i[t]$) is the KL-Divergence between the prior and posterior

$$V_i[t] = \mathcal{D}_{KL}(\mathbb{P}(f|\mu, \Sigma) \ || \ \mathbb{P}(f|\mu, \Sigma, \mathbf{S}_i[t])) \tag{2.5}$$

If the VoI for a particular agent exceeds a threshold V^*, then the agent broadcasts its local buffer $\mathbf{S}_i[t]$ to its neighbors and resets its buffer, or else it censors itself from the network. All agents relay messages received to their neighbors and update their prior with the relayed buffers. Given that at each time t all agents might not be informative and hence the communication between the agents is reduced resulting in a low communication cost.

A problem with a static VoI threshold V^* is a bounded error in estimate resulting from low communication frequency as $t \to \infty$. To eliminate this, a dynamic V^* is defined which has a higher values in the initial stages of communication since most agents are informative and decreases in later stages of time as agents develop better understanding of the underlying function and start censoring themselves from the network. It is shown in [36] that an adaptive V^* results in a continuous reduction in error in estimation. The AVoIDs algorithm is shown in algorithm 2 The communication cost in the network is defined as follows: Let indicator function $\mathbb{1}_{VoI}$ denote if agent i is informative at time t, then a communication cost function $\mathcal{C}[t]$ is defined as

$$\mathcal{C}[t] = \sum_{i \in \mathcal{V}} N \mathbb{1}_{VoI} \tag{2.6}$$

Where N is the cardinality of the $\mathcal{V}$. The adaptive VoI based information sensing algorithm is described in algorithm 2.

Algorithm 2 Adaptive Value of Information for Distributed Sensing(AVoIDs) [36]

```
1: procedure INITIALIZE HYPERPARAMETERS $\mu_i[0]$, $\Sigma_i[0]$, DEFINE $c^*$, $\gamma_1$, $\gamma_2$
2:     for each time $t = 1, 2, \ldots, T$ do
3:         for each agent $i \in \mathcal{V}$ do
4:             $\mu_i[t] = \mu_i[t-1]$
5:             $\Sigma_i[t] = \Sigma_i[t-1]$
6:             Take measurements $y_i$ and store in local buffer $\mathbf{S}_i$
7:             Calculate the Value of Information $V_i[t]$ using equation (2.5)
8:             if $V_i[t] \geq V^*$ then
9:                 Broadcast the buffer $\mathbf{S}_i$ and update prior distribution $p(f \mid \mu_i[t], \Sigma_i[t])$
10:                Reset buffer $\mathbf{S}_i = \mathbf{0}$
11:            end if
12:            Relay each received messages to neighbors
13:            for each broadcast message $\mathbf{S}_j$ do
14:                Update the posterior with $\mathbf{S}_j$
15:                $p(f \mid \mu_i, \Sigma_i, \mathbf{S}_j)$
16:                For each neighbor receiving message from $i$, increment communication cost
17:                $C[t] = C[t] + \bar{c}_i; \forall j \in N(i)$
18:            end for
19:            Adaptively change $V^*$
20:            if $C[t] < c^*$ then
21:                Too little communication
22:                $V^*[t+1] = \gamma_1 V^*[t]; (0 < \gamma_1 < 1)$
23:            else
24:                Too much communication
25:                $V^*[t+1] = \gamma_2 V^*[t]; (\gamma_2 > 1)$
26:            end if
27:        end for
28:    end for
29: end procedure
```

2.3 Gaussian Process Decentralized Data Fusion (GPDDF)

A third comparison method presented is the Gaussian Process Decentralized Data Fusion(GPDDF) algorithm proposed by Chen et.al. [11]. In GPDDF, each agent constructs a local summary based on measurements y_{A_k} available from its locations $A_k \in \mathcal{X}$ and communicates this summary model to its neighbors. The agent then updates the received summaries into its local summary to generate a global consistent summary, which is used to realize predictions over all locations in the domain $\mathcal{X}$. The local and global summaries are realized based on a set of common support set $U \subset \mathcal{V}$ defined *a-priori* for all agents. Hence given the support set U, a set of observed locations A_k with their corresponding measurements y_{A_k}, the local summaries for agent $i; \forall i \in \mathcal{V}$, $(\acute{y}^k_U, \acute{\Sigma}^k_{UU})$ are defined as

$$\begin{aligned} \acute{y}^k_U &= \Sigma_{UA_k} \Sigma^{-1}_{A_k A_k | U} (y_{A_k} - \mu_{A_k}) \\ \acute{\Sigma}^k_{UU} &= \Sigma_{UA_k} \Sigma^{-1}_{A_k A_k | U} \Sigma_{A_k U} \end{aligned} \tag{2.7}$$

The local summaries in equation (2.7) of each agent are exchanged with every other agent in the network i.e. $\forall i \in \mathcal{V}, N(i) = \mathcal{V} \setminus i$ and each agent $j \in N(i)$ assimilates every local summary into a global summary $(\tilde{y}_U, \tilde{\Sigma}_{UU})$ as

$$\tilde{y}_U = \sum_{i=1}^{N} \acute{y}^k_U \tag{2.8}$$

$$\tilde{\Sigma}_{UU} = \Sigma_{UU} + \sum_{i=1}^{N} \acute{\Sigma}^k_{UU} \tag{2.9}$$

The global summaries are used to predict function value in locations where no measurements are available. To improve the predictive capability the predictive mean and covariance are augmented using local observations

$$\begin{aligned} \mu^k_s &= \mu_s + (\gamma^k_{sU} \tilde{\Sigma}^{-1}_{UU} \tilde{y}_U - \Sigma_{sU} \Sigma^{-1}_{UU} \acute{y}^k_U) \\ \sigma^k_{ss'} &= \sigma_{ss'} - (\gamma^k_{sU} \Sigma^{-1}_{UU} \Sigma_{Us'} - \Sigma_{sU} \Sigma^{-1}_{UU} \acute{\Sigma}^k_{Us'} - \\ &\gamma^k_{sU} \tilde{\Sigma}_{UU} \gamma^k_{Us'}) - \acute{\Sigma}^k_{ss'} \end{aligned} \tag{2.10}$$

where γ^k_{sU} is defined as

$$\gamma^k_{sU} = \Sigma_{sU} + \Sigma_{sU} \Sigma^{-1}_{UU} \acute{\Sigma}^k_{UU} - \acute{\Sigma}^k_{sU} \tag{2.11}$$

GPDDF has been demonstrated empirically in modeling Mobility-on-Demand systems [11] and it has been shown to be more time-efficient and scalable with size of data in comparison with modeling full-GP. However in terms of communication it performs badly as at each time step agents are required to communicate entire information vectors and matrices containing the local summaries. The summary information increases in size with the number of support vectors used and hence in huge spaces with large numbers of agents operating, GPDDF is highly communication-inefficient.

The GPDDF algorithm is described in algorithm 3

Algorithm 3 Gaussian Process Decentralized Data Fusion (GPDDF) [11]

```
1: procedure A-PRIORI SUPPORT SET $U$, NETWORK STRUCTURE $\mathcal{G}$
2:     for each time $t = 1, 2, \ldots, T$ do
3:         for each agent $i \in \mathcal{V}$ do
4:             Construct local summaries using equation (2.7)
5:             Exchange local summaries with neighbors $j \in N(i)$ and obtain their summaries
6:             For each neighbor receiving a local summary from $i$, increment communication cost
7:             $C[t] = C[t] + \bar{c}_i; \forall j \in N(i)$
8:             Construct global summaries from relayed local summaries using equation (2.8)
9:             Predict function over $\mathcal{X}$
10:        end for
11:    end for
12: end procedure
```

Chapter 3

Gaussian Processes Information Fusion

In order to perform inference on the unknown function in a communication efficient manner while accounting for rumor propagation and duplication of information, and without having to make the common support assumption [11], we propose a novel algorithm labeled *GPFusion*. In GPFusion, sparse generative GP models are constructed from the measurement samples using an online sparsification technique [15] and these sparse GPs are broadcast by the agent to its neighbors. These sparse GP models can be considered as a compressed representations of the sampled measurements and therefore a significant reduction in communication can be achieved by transmitting these sparse GPs instead of the entire set of sampled measurements. Furthermore, not all agents have sufficiently changed local GP models at all times to contribute to the network. Therefore, in order to further improve communication efficiency, a censoring strategy inspired from [36] is used, which prevents agents from broadcasting models to their neighbors in the network, when their VoI does not exceed an adaptively adjusted VoI threshold.

In GPs the predictive covariance is good measure of confidence in an agent's prediction at any location in X [2, 26]. In the next subsection we describe how the problem of duplication of information and rumor propagation is tackled through an informed sampling strategy based on GP predictive covariance.

3.1 Description of the GP Fusion (GPF) Algorithm

In Gaussian Process Fusion (GPF) all agents maintain a local GP prior over the unknown function f. The support for GPs of each agent are not a-priori assigned, but rather are inferred from the data. Each agent $i \in \mathcal{V}$ samples a set of measurements $\bar{y}^i = \{y_{i1}, y_{i2}, \ldots, y_{i,p}\}$ at its location $x_i \in \mathcal{X}$. The measurements follow a likelihood function $\mathbb{P}_i(\bar{y}^i \mid f)$ and each agent i constructs a Gaussian Process model $\mathscr{GP}_i$ on f over domain $\mathcal{X}$. We abuse notation here and denote the coefficients of the mean and kernel functions for the sparse GP of agent i as β_i and C_i respectively, and coefficients of the composite GP due to fusion of model i into j as β_{ij} and C_{ij}. The predictive mean and covariance at any locations $x^* \in \mathcal{X}$ are then given by (3.1) and (3.2).

$$m_i = \beta_i^T k_{x^*} \tag{3.1}$$

$$\Sigma_i = k^* - k_{x^*}^T C_i k_{x^*} \tag{3.2}$$

Here $\beta_i = \beta_i(\bar{y}^i)$ and $C_i = C_i(\bar{y}^i)$ are functions of measurements $\bar{y}_i$ sampled by agent i(Refer to [15] for further details) and $k_{x^*} = k(x^*, \mathscr{BV}_i)$; where $\mathscr{BV}_i$ is the basis vector set of agent i's GP model.

To improve communication efficiency each agent $i \in \mathcal{V}$ transmits its learned parameters β_i, C_i and $\mathscr{BV}_i$, so that the GP model can be reconstructed at its neighbors $j \in N(i); \forall j$, instead of communicating measurement set $\bar{y}^i$. We leverage the property that GPs are generative models of data, that is samples from a GP recreate the dataset that they learned on its distribution. Therefore, agent's neighbors can recreate the data distribution, over the domain $\mathcal{X}$, by sampling the transmitted model at every point $x \in \mathcal{X}$. Let $\tilde{y}^{ij}$ represent the data of agent i recreated at its neighbor $j \in N(i)$. These measurements, which we label as *artificial measurements*, follow a likelihood function $\mathbb{P}(\tilde{y}^{ij} \mid m_i, \Sigma_i)$ and agent j can update its current model with these samples to obtain a composite model representing information from both agents. However, naively trusting the recreated measurements can lead to rumor propagation, especially if the neighboring agent has not seen sufficient data to enable its GP to make a confident prediction. To address this issue, we introduce an informed sampling strategy in which agent j samples the GP model of i at locations where agent i has confidence above a certain threshold σ_1, and agent j has confidence below another threshold σ_2. The advantage of such a definition is, the algorithm limits uncertain estimates of agent i from passing into model of j along with restricting measurements at intersecting regions of both agents from being double counted. The thresholds (σ_1, σ_2) determine the weight of information passed from i into j and proper selection of these values are important for the correctness of the algorithm. For example, setting σ_1 higher would allow more mistakes to propagate in the network whereas setting it low would prevent necessary information contained in i from passing into j.

To calculate the posterior fused mean, the local measurements at j, $\bar{y}^j$ and the artificial measurements $\tilde{y}^{ij}$ are combined into $\hat{\bar{y}}^j = [\bar{y}^j \;\; \tilde{y}^{ij}]$ and agent j's GP model $\mathscr{GP}_j$ is updated. Leveraging a result from [21], we show in Theorem 1 that since samples $\tilde{y}_{ij}$ are picked from locations in X where $\mathscr{GP}_i$ has high confidence, the samples lie within an small neighborhood of f with high probability. The GPFusion update on agent j results in a Gaussian Process with a mean and covariance functions linearly spanned by an updated set of basis vectors $\mathscr{BV}_{ij}$ and the updated coefficients β_{ij} and C_{ij}. The posterior mean and variance of the composite GP is given as :

$$\hat{m}_{ij} = \beta_{ij}^T k_{x^\star} \tag{3.3}$$

$$\Sigma_{ij} = k^* - k_{x^*}^T C_{ij} k_{x^*} \tag{3.4}$$

The updated coefficients are functions of $\hat{\tilde{y}}^j$ i.e. $\beta_{ij} = \beta_{ij}(\hat{\tilde{y}}^j)$ and $C_{ij} = C_{ij}(\hat{\tilde{y}}^j)$. An illustrative example of GPFusion is described in the next subsection.

Repeatedly transmitting the models with no new information from the incoming data still results in unnecessary communication which will increase the communication cost. At each time step the cost of transmitting model for each agent is $C_i[t] \leq |\mathscr{BV}_i|^2 + 2|\mathscr{BV}_i|$ and the total cost $C[t] = NC_i[t]$; N is the cardinality of the vertices set. To prevent this, agents determine the VoI contained in their model with respect to previous transmission and avoid broadcasting when the VoI is below a threshold. The VoI threshold is adaptively changed according to the communication cost. For sparse online Gaussian processes $N = |\mathscr{BV}|$ is the cardinality of the basis vector set of the GP model $\mathscr{GP}_i$ for agent i at time t. Given two N dimensional multivariate normal distributions $\mathscr{N}_0(\mu_0, \Sigma_0)$ and $\mathscr{N}_1(\mu_1, \Sigma_1)$ the KL divergence $(\mathcal{D}(\mathscr{N}_0 \| \mathscr{N}_1))$ between two multivariate Gaussians is given as

$$\begin{aligned}&\mathcal{D}(\mathscr{N}_0 \| \mathscr{N}_1) = \\ &\frac{1}{2}\{tr(\Sigma_1^{-1}\Sigma_0) + (\mu_1 - \mu_0)^\intercal \Sigma_1^{-1}(\mu_1 - \mu_0) - K + ln\frac{|\Sigma_1|}{|\Sigma_0|}\}\end{aligned} \tag{3.5}$$

The KL divergence between two Gaussian Processes can be approximated using (3.5) with the predictive mean and variance for all points in X, since Gaussian processes are infinite dimensional extensions of multivariate normal distributions [26].

It is desirable to regulate the cost $\mathcal{C}[t]$ around a reference value determined by the available communication bandwidth. If $\mathcal{C}[t] < \bar{c}$, the available communication bandwidth is under utilized, so the algorithm decreases $V^*[t]$ to encourage communication by setting $V^*[t+1] = \beta V^*[t+1]$; $0 < \beta < 1$. If $\mathcal{C}[t] \geq \bar{c}$, the communication cost is higher than desired, so the algorithm increases V^* to limit communication by setting $V^*[t+1] = \alpha V^*[t]$; $\alpha > 1$. The complete GPF algorithm is described in Algorithm 4.

Algorithm 4 Gaussian Process Fusion with Adaptive Value of Information for Distributed Sensing(GP-AVoIDs)

```
1: procedure GP PARAMETERS, NETWORK $\mathcal{G}, c^*, \sigma_1, \sigma_2$
2:     for each time $t = 1, 2, \ldots, T$ do
3:         for each agent $i \in \mathcal{V}$ do
4:             Take measurements $y^i = (y_1, y_2, \ldots, y_p)$ and update GP parameters according to (3.1)
   and (3.2)
5:             $m_i = \beta_i^T k_{x^*}$
6:             $\Sigma_i = k^* - k_{x^*}^T C_i k_{x^*}$
7:             Calculate Value of information between prior and posterior models using 3.5
8:             if $V_i[t] \geq V^*$ then
9:                 Transmit model parameters of $\mathscr{GP}_i$ to neighbors $j \in N(i)$
10:                Increment communication cost
11:                $C[t] = C[t] + \bar{c}_i[t]$;
12:            end if
13:            Get models from neighbor $j$ and integrate into current model
14:            Sample $\mathscr{GP}_j$ at locations where $\Sigma(f)_j \leq \sigma_1$ and $\Sigma(f)_i \geq \sigma_2$ to obtain $\tilde{y}^{ij}$1s
15:            Update $\mathscr{GP}_j$ with artificial measurements $\hat{\tilde{y}}^j = [\bar{y}^j \ \tilde{y}^{ij}]$ and obtain updated model
16:            $\hat{m}_{ij} = \beta_{ij}^T k_{x^*}$
17:            $\Sigma_{ij} = k^* - k_{x^*}^T C_{ij} k_{x^*}$
18:            Adaptively change $V^*$
19:            if $C[t] < c^*$ then
20:                Too little communication
21:                $V^*[t+1] = \gamma_1 V^*[t]; (0 < \gamma_1 < 1)$
22:            else
23:                Too much communication
24:                $V^*[t+1] = \gamma_2 V^*[t]; (\gamma_2 > 1)$
25:            end if
26:        end for
27:    end for
28: end procedure
```

In the following theorem 1 its shown that the error in estimation between two GPs, one modeled with true measurements and other modeled with artificial measurements is bounded in error.

Theorem 1. *Consider a network of N autonomous agents which are inferring the underlying function f using algorithm 4. Suppose, for a given ϵ and δ, each agent generates artificial samples at locations x s.t.* $\sigma(x) < \frac{2\omega^2 a^2}{A^2 n\beta^2\Delta^2 log(\frac{2}{\delta})}$*, then the error in estimation is bounded in probability.*

Proof. Consider an agent i in network $\mathcal{G}$ estimating the unknown function f using 4. At time t assume agent i receives a model from j and generates artificial samples $\hat{\bar{y}}_i = [y_1, y_2, \ldots, \hat{y}_\alpha, \ldots, y_n]$ which are used to update the posterior distribution. If agent i uses the true samples $\bar{y} = [y_1, y_2, \ldots, y_\alpha, \ldots, y_n]$ to update the posterior, then the error between the two estimates can be calculated as

$$e(t) \quad = \sqrt{\sum_{\forall x \in \mathcal{X}} \left| f(x,t) - \hat{f}(x,t) \right|^2} \tag{3.6}$$

Taking expectation over the error will result in

$$\mathbb{E}[e(t)] \quad \leq \sqrt{\sum_{\forall x \in \mathcal{X}} \left| \mu_i(x,t) - \hat{\mu}_i(x,t) \right|^2} \tag{3.7}$$

Using the GP regression equations defined in equation (1.5), the right hand side of equation (3.7) can be written as

$$\begin{aligned} \mathbb{E}[e(t)] \quad &\leq \sqrt{\sum_{\forall x \in \mathcal{X}} \left| k^T C \bar{y}_i - k^T C \hat{\bar{y}}_i \right|^2} \\ &= \sqrt{\sum_{\forall x \in \mathcal{X}} \left| k^T C [\bar{y}_i - \hat{\bar{y}}_i] \right|^2} \end{aligned} \tag{3.8}$$

Using the Cauchy-Schwartz inequality on the above equation we obtain the following inequality

$$\mathbb{E}[e(t)] \quad \leq \sqrt{\sum_{\forall x \in \mathcal{X}} \left\| k^T C \right\|_2 \left\| [\bar{y}_i - \hat{\bar{y}}_i] \right\|_2} \tag{3.9}$$

In equation (3.9) the difference $\left\| [\bar{y}_i - \hat{\bar{y}}_i] \right\|_2$ is independent of x and hence can be defined outside the summation, reducing the expectation to

$$\mathbb{E}[e(t)] \quad \leq \left[\sqrt{\sum_{\forall x \in \mathcal{X}} \left\| k^T C \right\|_2} \right] \left[\sqrt{\left\| \bar{y}_i - \hat{\bar{y}}_i \right\|_2} \right] \tag{3.10}$$

By applying Markov's inequality on the random variable $e(t)$ we obtain the bound as

$$\begin{aligned}
\mathbb{P}(e > a) &\leq \frac{\mathbb{E}[e]}{a} \\
\implies \mathbb{P}(e > a) &\leq \frac{\left[\sqrt{\sum_{\forall x \in \mathcal{X}} \left\|k^T C\right\|_2}\right]\left[\sqrt{\left\|\bar{y}_i - \hat{\bar{y}}_i\right\|_2}\right]}{a} \\
\implies \mathbb{P}(e > a) &\leq \frac{A\left[\sqrt{\left|\left\|\bar{y}_i - \hat{\bar{y}}_i\right\|_2\right|}\right]}{a}
\end{aligned} \tag{3.11}$$

where $A = \left[\sqrt{\sum_{\forall x \in \mathcal{X}} \left\|k^T C\right\|_2}\right]$

Since the artificial samples are obtained from a model generated by the true samples $\bar{y}_i$, it can be claimed that $\hat{\bar{y}}_i \sim \mathbb{P}(\bar{y}_i \mid \mathcal{X})$.

From the corollary of the McDiarmid's Theorem discussed in [21], the mistake in the posterior prediction made at location $x \in \mathcal{X}$ is bounded in probability if the variance $\sigma(x)$ at location x has a variance below a specific threshold i.e $\mathbb{P}(\left|f(x) - \mu(x)\right| \leq \epsilon) \geq \delta$ if $\sigma(x) < \frac{2\omega^2\epsilon^2}{\Delta^2 log(\frac{2}{\delta})}$; where $\Delta = y_{max} - y_{min}$.

In order to ensure that error is bounded, the difference $A\left[\sqrt{\left\|\bar{y}_i - \hat{\bar{y}}_i\right\|_2}\right]$ is smaller than a and this can be ensured by selecting locations x, which selects $\hat{\bar{y}}_i$ ensuring $\left\|\bar{y}_i - \hat{\bar{y}}_i\right\|_2$ is small. If ϵ is the deviation in artificial estimate from the true sample then $\left\|\bar{y}_i - \hat{\bar{y}}_i\right\|_2 = \sqrt{n}\epsilon$. The equation (3.11) reduces as

$$\mathbb{P}(e > a) \quad \leq \tfrac{A\sqrt{n}\epsilon}{a} \tag{3.12}$$

The probability holds if ϵ is selected such that

$$\epsilon \leq \frac{a}{A\sqrt{n}\beta} \tag{3.13}$$

where β is chosen to be a large value, such that the probability has a tight upper bounded. Then the variance is selected at points where

$$\sigma(x) < \frac{2\omega^2\epsilon^2}{\Delta^2 log(\frac{2}{\delta})} < \frac{2\omega^2 a^2}{A^2 n\beta^2\Delta^2 log(\frac{2}{\delta})} \tag{3.14}$$

Selecting points at x below the variance ensures that error in network due to communicating models is bounded.

■

3.2 An Illustrative Example

We demonstrate the idea behind GPFusion with a simple example in which two agents learn a sine function in domain [-1,1], when agents are constrained spatially in sampling the function. The agents, A1 and A2 are spatially separated and constrained to sample f in local neighborhoods $D_1 \neq D$ and $D_2 \neq D$. Figure 3.1 depicts the setup, where the sensing region of A1 D_1 is shown with the red points, and the sensing region of A2 D_2 is shown with the black points. Note that the sensing domains of the two agents intersect. Figure 3.2a shows the GP estimate of the unknown function by A1 in D. In the figure, the green band shows the agent's predictive variance; higher predictive variance means that the agent has little confidence in its predictions and vice-versa. As shown in the Figure 3.2a, A1 has high confidence in the region D_1 (as indicated by its low predictive variance) since it has measurements from this region and has a low confidence in $D \setminus D_1$ (indicated by high predictive variance) since it has no measurements in that region. When A1 receives A2's model, A1 samples the model in domain D_2 where the variance of A2 is below threshold σ_1 and where A1's own variance is above σ_2 as explained in algorithm 4. The points sampled by A1 are shown by the blue points in figure 3.2a. Given sufficient budget, both agents obtain a good estimate of the unknown function without sampling f over the entire domain as indicated by Figures 3.2c and 3.2d, as each agent incorporates the information from the artificial measurements derived from their neighbors models.

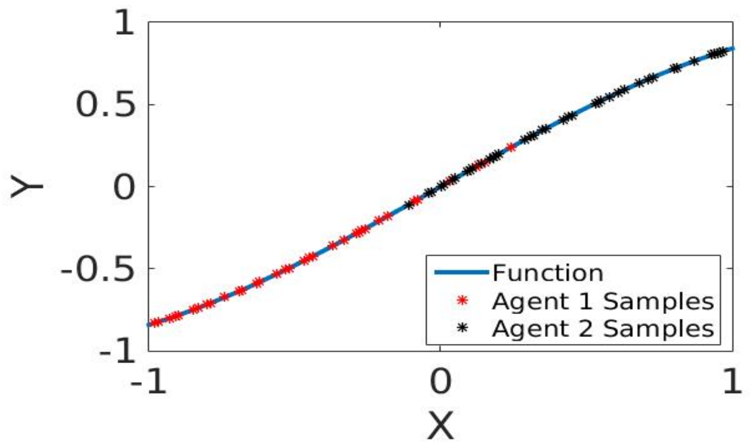
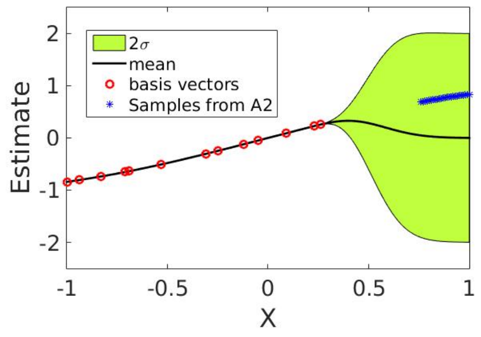
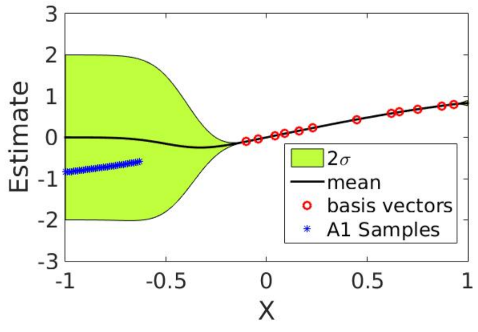


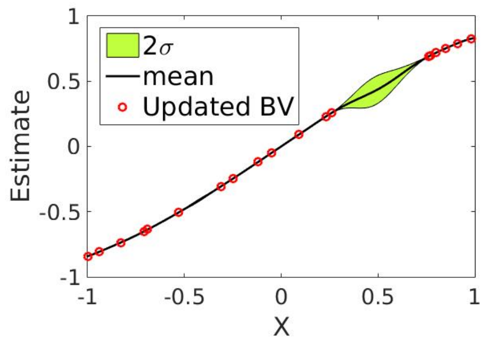
Figure 3.1: Spatially separated agents sampling an unknown function. The red points indicate the operating domain of A1 and black points indicate the operating domain of A2



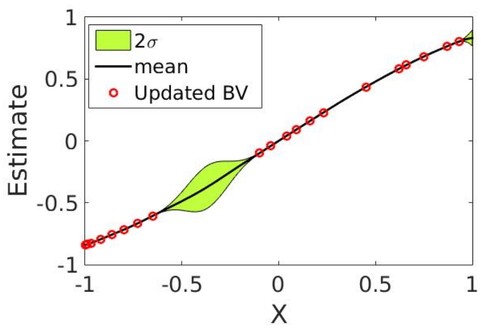
(a) Agent 1 picking samples from estimate of Agent 2



(b) Agent 2 picking samples from estimate of Agent 1



(c) Agent 1 estimate after fusion update



(d) Agent 2 estimate after fusion update

Figure 3.2: 1D example demonstrating GP Fusion procedure. The example demonstrates two agents converging to an accurate representation when sections of the function are inaccessible

Chapter 4

Experimental Validation on Synthetic and Real-World Datasets

To test the communication efficiency of the GPFusion class algorithms in comparison with our presented baseline methods (Weighted Average Consensus, AVoIDs, and GPDDF, see section 2 for details), data from two real world datasets and one synthetic dataset is collected, and passed through a MATLAB simulation of the sensor network. The MATLAB simulations were run on an Intel i7 Processor with 16 GB DDR3 ram.

Comparisons in error to a centralized estimate(which is assumed to capture the true function), with final communication cost are studied for all algorithms discussed in the paper, and the results indicate improvement in both estimation and communication efficiency with GPFusion methods.

4.1 Evaluation on Synthetic Dataset

A dataset indicating air density in a room was synthesized on a 50×50 grid world domain and sensors were placed at random locations to sample the density values from the air density function shown in figure 4.1.

The agent locations are selected randomly and the network is connected in a ring topology, which is a network type where agents are connected to exactly two nodes forming a continuous pathway. In ring topology data travels along the network with every node handling the data satisfying the conditions of strong connectivity. The sensor agents receive measurements $\bar{y}_i$ from the function corrupted with a small amount of Gaussian noise ν i.e $\bar{y}_i \sim p(D \mid f)$, where $p(D \mid f) = \mathcal{N}(f, \nu)$. The simulation is run for a total of 100 seconds where the model parameter of the underlying function is constant.

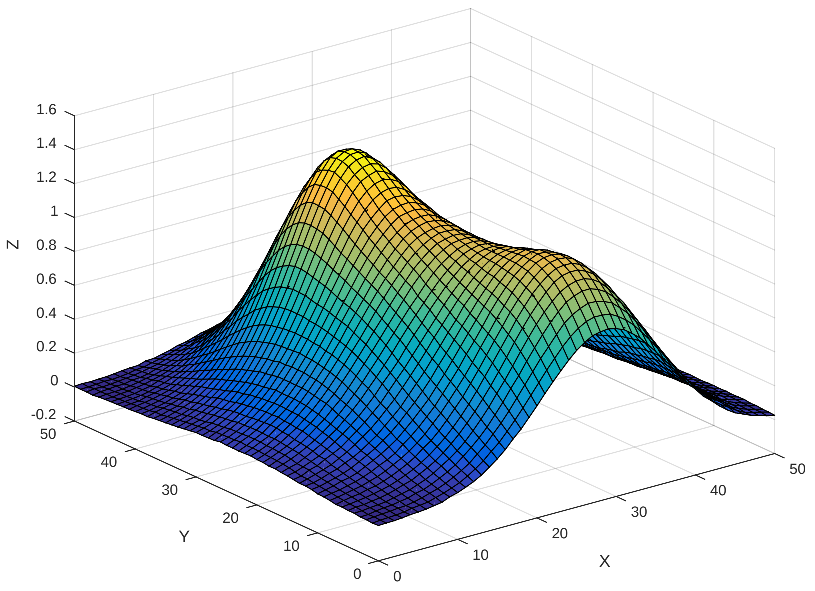


Figure 4.1: Synthetic air density on 50×50 gridworld

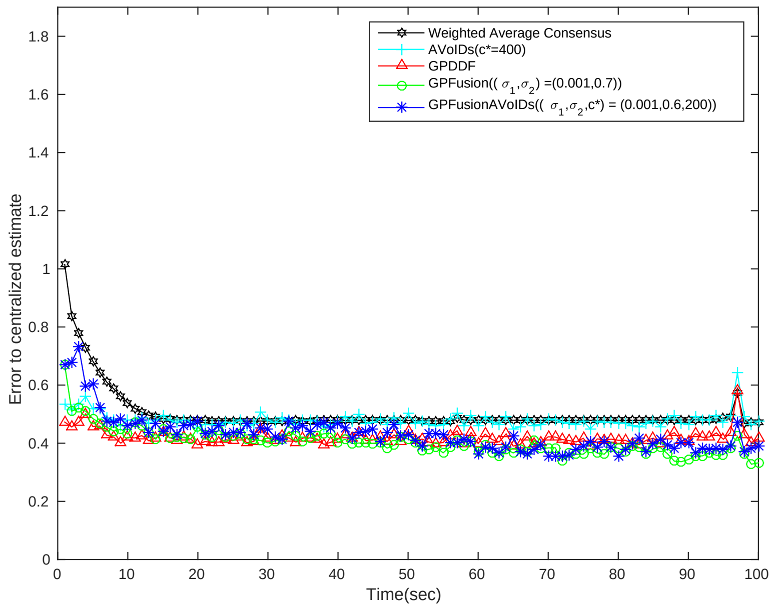


Figure 4.2: Error to centralized estimate with time

Figure 4.2 shows the error for each method described, as a function of time. As the plots indicate, Weighted Average Consensus(WAC) starts with a high error but exponentially converges to a steady value as all agents update model parameters and reach an average value. AVoIDs has a similar error performance to WAC since the underlying model fusion is a Bayesian filtering algorithm which updates the parameter vector. The error in the function estimated using GPDDF algorithm maintains a nearly constant value as time progresses once agents converge to the parameter values. But since the algorithm maintain a fixed basis vector set, agents cannot place a kernel at a location where high information can be obtained. This is the advantage with GPFusion class algorithms which start with higher errors, but as time passes they show exponential reduction in error till a steady error value is reached. Since the GPFusion methods rely on online sparse Gaussian Processes which select kernel locations using the kernel independence test (refer to section 1.2 or [15] for details), the support vectors will adaptively change based on the incoming measurements.

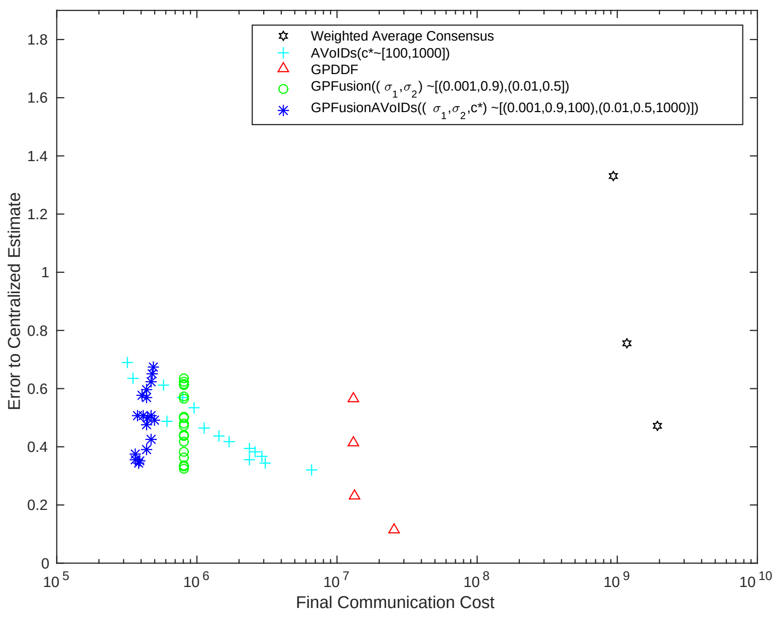


Figure 4.3: Error to centralized estimate vs final communication cost for the synthetic density dataset. GPFusion class algorithms lies in the bottom left region showing good accuracy along with lower communication costs.

Figure 4.3 shows cost and error performance on error-cost coordinates. The x-axis represents the cost coordinates, which represent the final cost incurred by the algorithm at the end of the simulation and the y-axis shows mean error of all agents to the centralized estimate. The centralized estimate is modeled by passed all sampled data to a central location where a full GP is trained on the measurements. It is desirable for a communication efficient algorithm to lie in the bottom left corner of the plot with low cost and error.

Weighted Average Consensus lies towards the right corner with high cost and error which indicates that it performs poorly in both cost and error. The data points are generated by using a total of 40, 45 and 50 agents in the network. This is inherent in consensus type algorithms since agents update their model parameters to move closer to their neighbors and any error in model estimation by an agent is passed to all agents in the network.

GPDDF is also controlled by changing the number of agents in the network. Better estimation is shown by increasing the number of agents in the network. The plots show that final cost and error lies in the lower right corner, which indicates good accuracy in estimation, but a high communication cost.

AVoIDs, GPFusion and GPFusion-AVoIDs are simulated with a total of 50 agents in the network. The control parameters for the cost and error are the thresholds c^* for AVoIDs and σ_1, σ_2 and, c^* for GPFusion and GPFusion-AVoIDs.

GPFusion with threshold values of $(\sigma_1, \sigma_2) = (0.001, 0.5)$ has higher error to centralized estimate since the lower variance threshold of σ_2 doesn't allow propagation of higher number of samples from neighbors models. The error however reduces by properly tuning the values of σ_1 and σ_2. The communication cost remains a low constant value, since the overall number of budget vectors exchanged during the duration is upper bounded. GPFusion-AVoIDs has the lowest communication cost in comparison to other methods as it has the advantages of GPFusion where information is propagated with a small set of basis and weights and the cost based censoring of the agents on the value of information. The general trend is GPFusion-AVoIDs performs well in terms of both accuracy of estimation and communication cost.

4.2 Evaluation on EDGAR Methane Emissions Data Set

The first set of real world data tests were performed on methane emissions data obtained from Europa Joint Research Center's (JRC) Emission Database for Global Atmospheric Research (EDGAR). The EDGAR project [39] is a compilation of global emission data of green house gases covering a period of 38 years. The data is high resolution spatially detailed obtained from a satellite on a $0.1^0 \times 0.1^0$ grid. The data was compiled for individual countries based of emissions factors that are regional specific or international. A sampled subset of the total time series during the period of 2006-2008 over the region of Southeastern Oklahoma was selected. The emissions rate for methane are slow enough to assume rate of change in temporal distributions are constant with time over this period. Figure 4.4 shows one time frame of the methane distribution over the earth surface as an example.

The spatial area was gridded into a 50×50 grid world area and emissions data is defined over

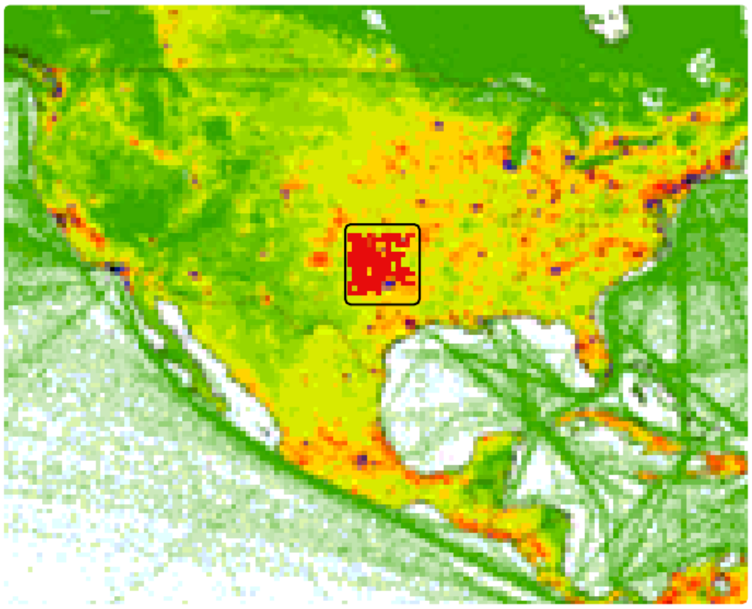


Figure 4.4: Distribution of methane(CH_4) over North America captured by the Europa's JRC satellite. The boxed region shows the region from which data is selected for conducting experiments.

this gridded locations. The network is connected in ring structure to ensure strong connectivity and agent positions are selected to maximize coverage of the region. The results from this dataset indicate the GPFusion class algorithms perform well even when the underlying parameters are slowly instead of assuming to be constant.

Figure 4.5 shows the error for each method on the EDGAR dataset with time. Weighted Average Consensus has the lowest accuracy with an error value of 0.3 units of emissions when a total of 50 agents are operating in the network. With the same number of agents GPDDF has a slightly better error estimate of 0.26 units of emissions. For AVoIDs with a cost threshold of $c^* = 200$, the error is better than GPDDF as a lower cost threshold causes agents to have higher communication frequency every so often leading to better estimation of the model parameters. GPFusion class algorithms have the lowest error in estimation. In GPFusion, the agents exponentially reduce their error to a steady value after approximately 25 time steps. In GPFusion-AVoIDs, the error reduced steadily over time and the error moves closer to the error of GPFusion. In both method the error is lower bounded based on the coverage agents have over the domain.

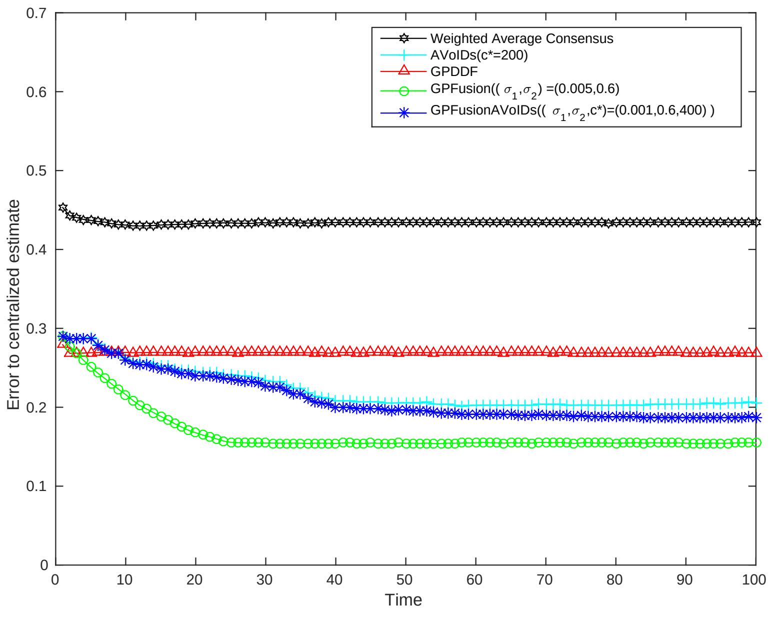


Figure 4.5: Error to centralized estimate with time on EDGAR dataset. GPFusion class methods show lowest error in comparison with other estimation methods.

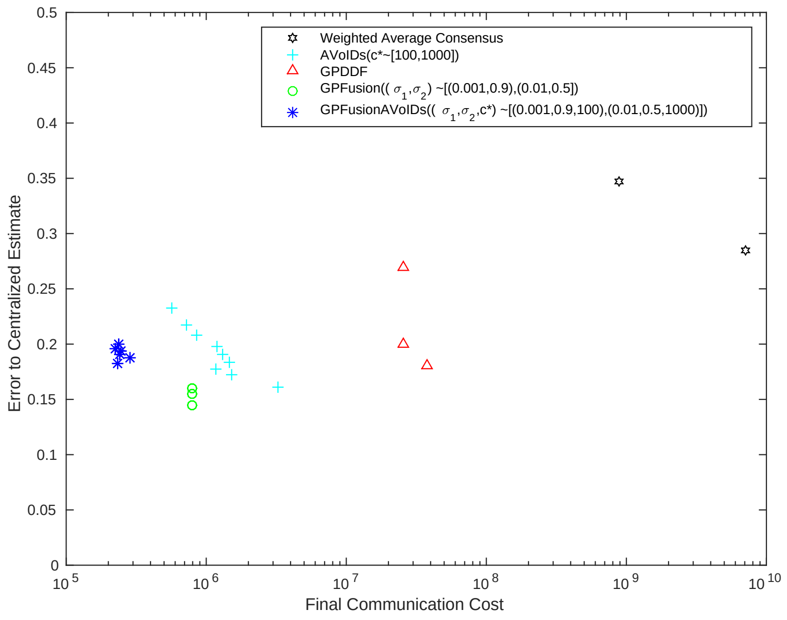


Figure 4.6: Error to centralized estimate vs final communication cost for the EDGAR Methane dataset. GPFusion class algorithms lies in the bottom left region showing good accuracy along with lower communication costs.

Figure 4.6 shows cost and error performance on cost-error coordinates. Weighted Average Consensus lies towards the right corner with high cost and error. GPDDF is also simulated by changing the number of agents in the network. Better estimation is shown by increasing the number of agents in the network. The plots show that final cost and error lies in the lower right corner, which indicates low error but high communication costs. Both GPFusion and GPFusion-AVoIDs lie in lower left corner with a lower error and communication cost indicating that even with slow changes in parameter GPFusion class algorithms have better performance.

4.3 Evaluation on Intel Berkeley Temperature Dataset

The Intel Berkeley dataset contains temperature, humidity, light flux and voltage data collected from 54 sensors arranged at Intel Berkeley research labs from Feb 28th 2004 to April 1st 2004. Mica2dot sensors with weatherboards were deployed at various locations over the lab and collected the above data once every 31 seconds. The arrangement of the sensors in the lab is shown in Figure 4.7.

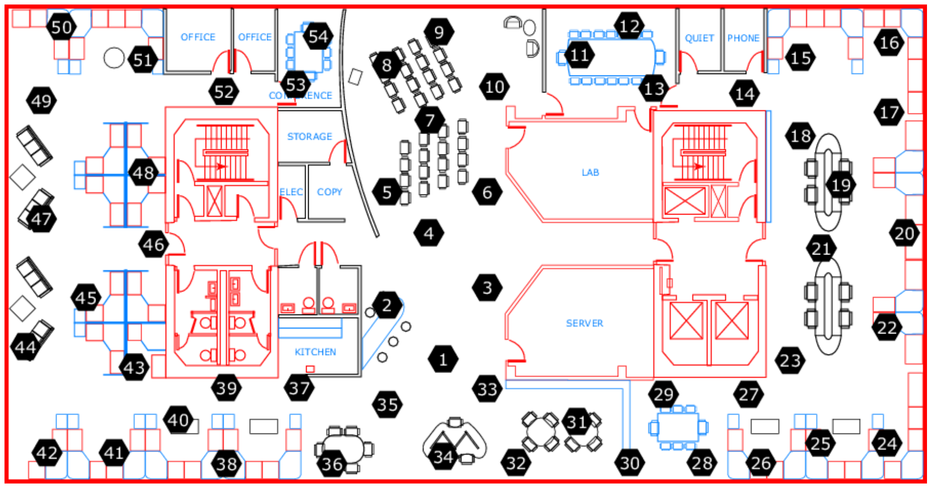


Figure 4.7: Arrangement of Sensor at Intel Berkeley

The algorithms were evaluated on the temperature dataset. The temperature dataset was shown to have small drifts during the day over small time intervals i.e. within range of $5°C$ and for the purpose of evaluating our algorithm, the change is parameter can be assumed to be small. Therefore the algorithm is evaluated by selecting data at a frequency of 20 minutes in a period of 28 hours during March 6 and 7 2004. The temperature variation for each agent for the selected time period is shown in figure 4.8.

To construct the centralized function from the data measurements a 45×35 grid world domain

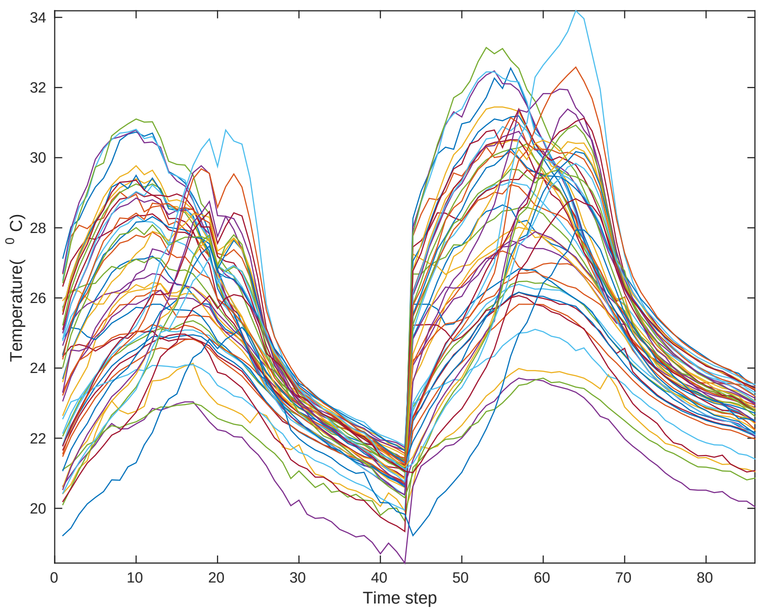


Figure 4.8: Temperature variation over 50 sensors used for the purpose of evaluating the algorithms

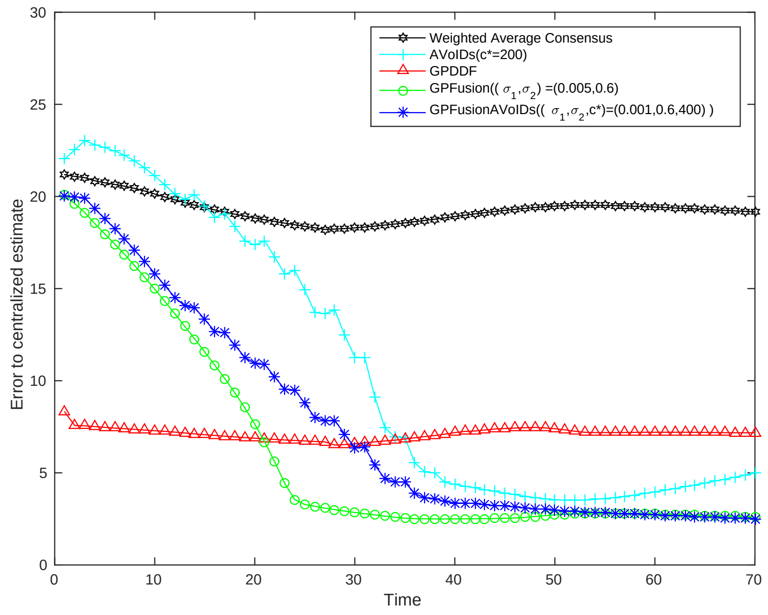


Figure 4.9: Error to centralized estimate with time

was simulated to approximate the lab environment and data was passed into the sensor network.

Figure 4.9 plots the error to centralized estimate as function of time. WAC is observed to have a very poor error estimation as underlying function is time varying and every time agents update their models, the underlying function has changed. GPDDF has better error estimation

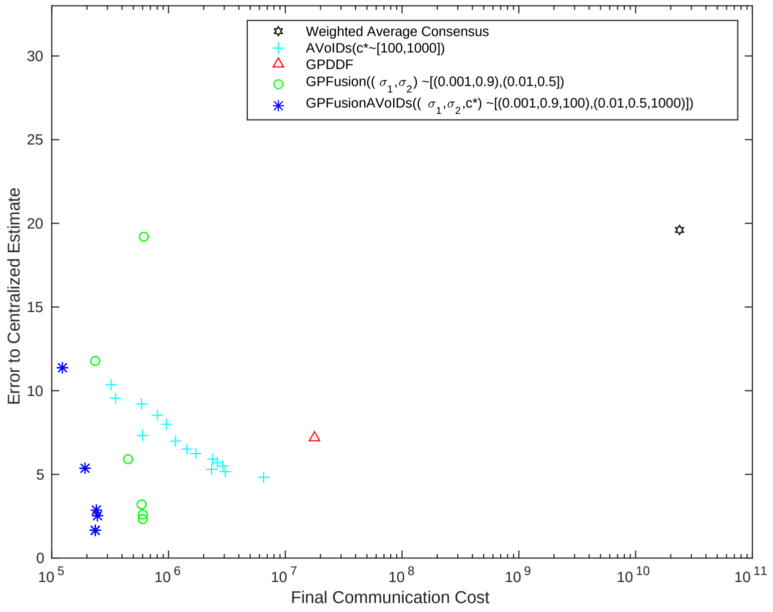


Figure 4.10: Error to centralized estimate vs final communication cost for the Intel Berkeley dataset. GPFusion class algorithms lies in the bottom left region showing good accuracy along with lower communication costs.

than WAC and maintains a steady error though the entire time period. As seen earlier for AVoIDs the error exponentially reduces lower than GPDDF. GPFusion class algorithms have the lowest error in estimation. In GPFusion, the agents exponentially reduce their error to a steady value. In GPFusion-AVoIDs, the error reduced steadily over time and the error moves closer to the error of GPFusion. In both method the error is lower bounded based on the coverage agents have over the domain.

In Figure 4.10 the plots show the relationship between cost and error. The final mean error for GPDDF is 8°C and communication cost is 2×10^7. AVoIDs has a max error of 11°C for a cost of 4×10^5 bits and a lower error of 6°C with a cost of 8.6×10^6 bits. The communication cost for GPFusion lies in a range of $4{\times}10^5$ and 5×10^5 and the minimum error to centralized estimate is 1.98°C. Communication costs for GPFusion-AVoIDs lie inside the range 1.02×10^5 and 2×10^5 with a minimum error of 1.97°C.

4.4 Conclusion

This chapter presented solution to the distributed inference problem for a spatially separated sensor network with constrained communication capabilities. GPFusion algorithm presents a novel way of combing two Gaussian Processes which is exploited to result in a significant reduction in communication cost with little need for a priori domain knowledge. In particular, it is shown to be better than other decentralized inference algorithms without having to assume that all agents share an a-priori agreed covariance structure. The approach discussed enables every agent to adapt the structure of its model to best suite its environment, while agents learn a global model through communication efficient local model fusion. The algorithm was validated one synthetic and two real world datasets where it was shown to lead to comparable performance with state-of-the-art methods, without having to assume a common set of kernels.

Chapter 5

Multi-Agent Game Emulator(MAGE)

This chapter discusses a multi-agent simulation framework to perform algorithmic research and proof-of-concept implementation before making use of delicate, expensive, and time-consuming real-world robotics systems.

Simulated environments have been vital in most robotics research. The need for a multi-agent UAS simulation environment is immediate considering the growing interest in collaborative autonomy. Planning under uncertainty for teams of unmanned aerial and ground systems is a widely studied topic in the field of artificial intelligence. In Dec-POMDP based multi-agent planning algorithms surveyed in [5], agents search for optimal policies over a number of scenarios using multiple models to learn a joint value function [4, 7]. The practical implementation of most of these methods has been hindered by the lack of high-fidelity multiagent simulators that can enforce realistic flight-relevant constraints. Additionally, the field of Human-Robot Interaction (HRI) and co-robotics deals with the design of effective tools for enabling humans and robots to solve complex tasks together, and has gathered wide research attention. Various methods studied include tele-autonomous control of robots [50], transfer learning between humans and robots [6, 18], and shared control methods [19]. However, searching for optimal policies in the presence of human-robot interaction in complex mission scenarios (see for example the elements of a DoD mission scenario in Figure 5.1) is a difficult task, partially because good models of human behavior in high stress situations are not always available.

In this chapter we provide a detailed description of the individual open source components from which the MAGE simulation environment is developed.

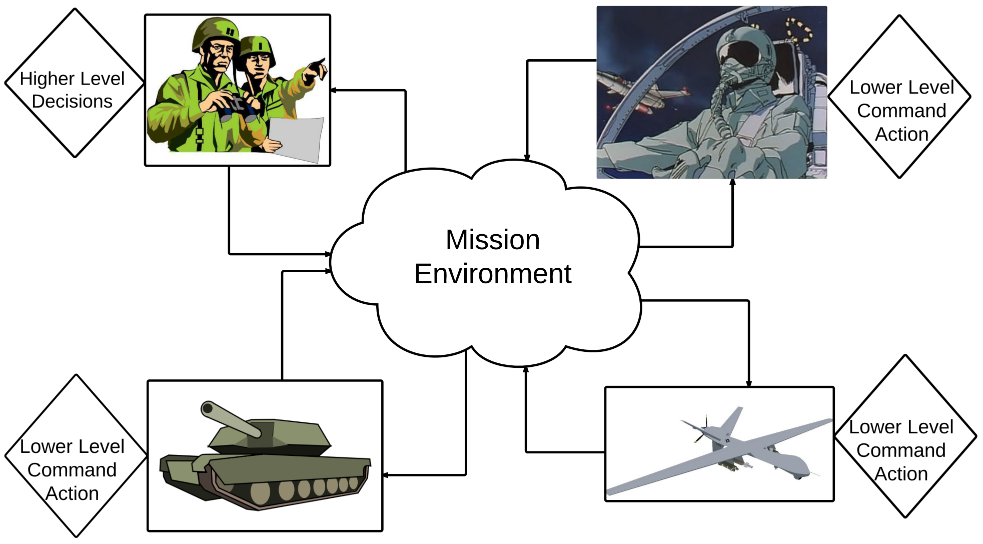

Higher Level Decisions
Lower Level Command Action
Mission Environment
Lower Level Command Action
Lower Level Command Action

Figure 5.1: Example of Human-Robot Interactions in a DoD mission. Manned and unmanned agents operate in a common mission environment and take decisions to maximize a mission score

5.1 Flightgear

Flightgear [42] is an open-source flight simulator framework that has been used in research, academics and even in pilot training. Flightgear aims to be portable across many different operating systems and offers a very flexible framework for developing extensions. The file formats are open and easily accessible. Standard 3D model formats are supported, and much of the simulator configuration is controlled through XML-formatted ASCII files which simplifies adding extensions to Flightgear without any major modifications to the source code. Flightgear also offers an internal scripting language called *Nasal* which supports reading and writing internal Flightgear properties, accessing internal data via extension functions which manipulate the source code.

The extension capabilities of Flightgear are made available via the "property tree", of the simulation infrastructure developed by SimGear [40]. The property tree, partly shown in Figure 5.2, is an interface to the runtime state variables of each vehicle in the simulation, allowing them to be obtained and manipulated during runtime. For example, the current longitude of the vehicle or the current aileron position can be accessed from the property tree in variables such as **/position/longitude-deg** and **/controls/flight/aileron** respectively. Thus many parameters such as FDM states, control, AI models, environment parameters, multiplayer participants and network states can be accessed and modified through extensions to Flightgear.

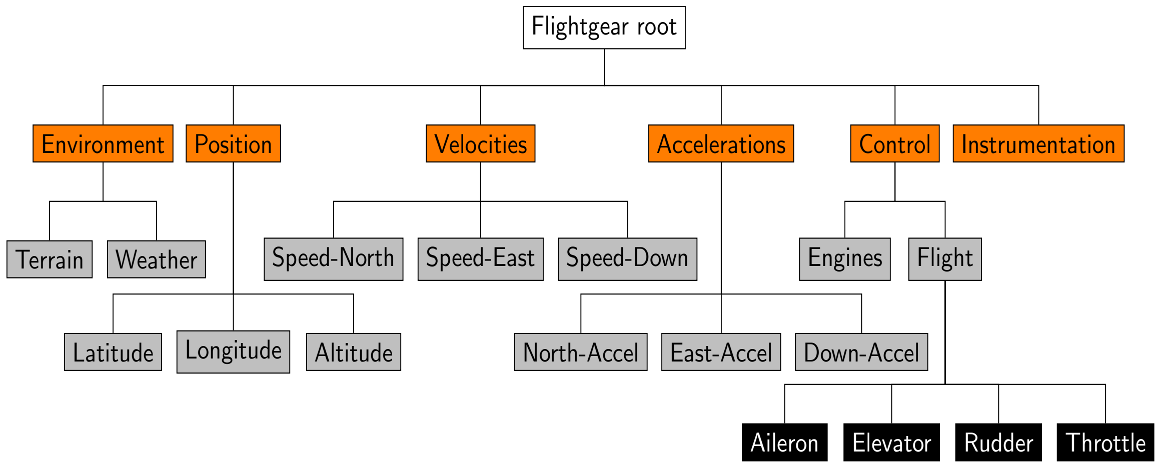
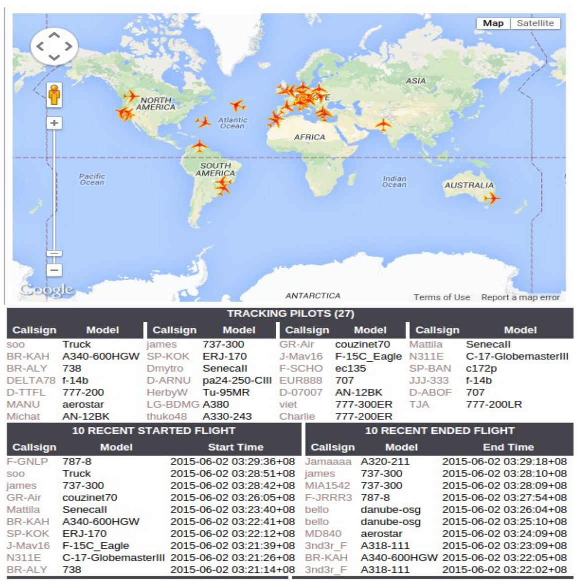


Figure 5.2: Flightgear's property tree structure, containing references to each vehicle's runtime state variables



TRACKING PILOTS (27)

Callsign	Model	Callsign	Model	Callsign	Model	Callsign	Model
soo	Truck	james	737-300	GR-Air	couzinet70	Mattila	Senecall
BR-KAH	A340-600HGW	SP-KOK	ERJ-170	J-Mav16	F-15C_Eagle	N311E	C-17-GlobemasterIII
BR-ALY	738	Dmytro	Senecall	F-SCHO	ec135	SP-BAN	c172p
DELTA78	f-14b	D-ARNU	pa24-250-CIII	EUR888	707	JJJ-333	f-14b
D-TTFL	777-200	HerbyW	Tu-95MR	D-07007	AN-12BK	D-ABOF	707
MANU	aerostar	LG-BDMG	A380	viet	777-300ER	TJA	777-200LR
Michat	AN-12BK	thuko48	A330-243	Charlie	777-200ER		

10 RECENT STARTED FLIGHT

Callsign	Model	Start Time
F-GNLP	787-8	2015-06-02 03:29:36+08
soo	Truck	2015-06-02 03:28:51+08
james	737-300	2015-06-02 03:28:42+08
GR-Air	couzinet70	2015-06-02 03:26:05+08
Mattila	Senecall	2015-06-02 03:23:40+08
BR-KAH	A340-600HGW	2015-06-02 03:22:41+08
SP-KOK	ERJ-170	2015-06-02 03:22:12+08
J-Mav16	F-15C_Eagle	2015-06-02 03:21:39+08
N311E	C-17-GlobemasterIII	2015-06-02 03:21:26+08
BR-ALY	738	2015-06-02 03:21:14+08

10 RECENT ENDED FLIGHT

Callsign	Model	End Time
Jamaaaa	A320-211	2015-06-02 03:29:18+08
james	737-300	2015-06-02 03:28:10+08
MIA1542	737-300	2015-06-02 03:28:09+08
F-JRRR3	787-8	2015-06-02 03:27:54+08
bello	danube-osg	2015-06-02 03:26:04+08
bello	danube-osg	2015-06-02 03:25:10+08
MD840	aerostar	2015-06-02 03:24:09+08
3nd3r_F	A318-111	2015-06-02 03:23:09+08
BR-KAH	A340-600HGW	2015-06-02 03:22:05+08
3nd3r_F	A318-111	2015-06-02 03:22:02+08

(a) Multiplayer map tracks all vehicles operating on the multiplayer servers

(b) Multiple aircraft interacting in a common scenario in Flightgear's multiplayer environment. Aircrafts can be initialized either on a single machine or multiple machines

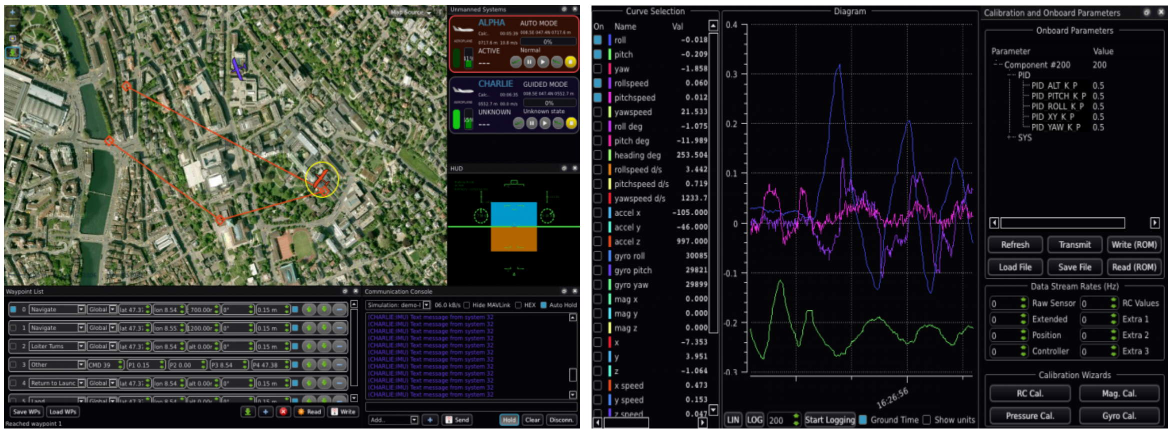
Figure 5.3: Flightgear Multiplayer System

Flightgear also supports multiple concurrent I/O connections. This permits an external program to read and write information from and to Flightgear, providing a mechanism to allow either an external controller or a human operator to control the vehicle inside the program. Flightgear allows this through two protocol definitions called the *FGNetFDM* protocol and *generic* protocol. Both protocols send and receive data from an external program using UDP packets over standard socket communication. The generic protocol allows defining custom packets with specified fields through a standard XML configuration file. A generic communication protocol can be easily defined for both input and output and allows for data exchange with the external program via sockets, files or First In, First out (FIFO) process queues. In the current simulator design we interface with Flightgear's property tree using a generic protocol over the network. The properties pulled and manipulated from the property tree are the standard state variables of position, velocity, acceleration and PWM control. When a human operator is controlling the vehicles the MAGE simulator launch system allows the user to switch the control to a keyboard or a joystick. Further details on the simulator launch system are provided in section 5.4.

Another attractive feature of Flightgear is its multiplayer capabilities [9]. The multiplayer feature allows vehicles to interact with each other, making it possible to perform collaborative airborne operations such as flying in formation, airport traffic management, or engaging in dog-fight-like maneuvers. Multiple Flightgear instances can be launched on different workstations and can interact using a LAN (Local Area Network) or the Internet in peer-to-peer connection or through a public host. The Flightgear Multiplayer server is a standalone network server that allows interactions between various Flightgear instances connected on an network. Figure 5.3a shows the multiplay map, which is a tracking system for vehicles operating on Flightgear's multiplayer servers. The system tracks vehicles launched from multiple Flightgear instances, connected to a public server over the Internet. The Multiplayer server is built upon the Multiplayer protocol which defines the message format for information exchange between Flightgear instances in the network. Each message packet consists of a header and block of data which are encoded in XDR encoding format. The encoded messages are sent via the UDP protocol. The header is a 32-byte message and contains the message ID, message length, reply address, reply port and player callsign fields. The multiplayer system can be launched either on a private or a public domain.

5.2 QGroundControl

QGroundControl is an open source aerial vehicle ground control unit whose capabilities include 2D and 3D maps with drag-and-drop waypoint capabilities and real-time plotting of sensor and telemetry data. Figure 5.4 shows the GUI interface for QGroundControl. Figure 5.4a shows the waypoints followed by two aircraft on a 2D map visualization. Figure 5.4b shows the monitoring



(a) Waypoints on 2D Earth Visualization (b) Telemetry Data Display

Figure 5.4: Flightgear Multiplayer System

windows for one of the aircraft during its flight. It is a modular application whose architecture is optimized for future extensions and open-source contributions. QGroundControl's main interface is MAVlink, a binary serial stream protocol which QGroundControl can receive over UDP.

QGroundControl is an object oriented C^{++}/Qt application that allows representation and visualization of aerial vehicles. The capabilities can be extended to include ground vehicles by adding new custom physical links and protocols conforming with designated unit conventions. QGroundControl complies with ISO-defined Open System Interconnections (OSI). This means that the data, data manipulation, and user interface are separated.

QGroundControl works primarily with the MAVLink protocol [32] which offers two-way datalink between QGround Control and the UAV by packing data into C structures that can store data very efficiently. Once the packet is formed, it can be sent from the UAV to QGroundControl as a byte sequence that contains the UAV's position, attitude, etc. When used to interact with physical robots, the link is maintained with radio transmitters. However, QGroundControl supports a variety of data protocols/connections/sockets (UDP, TCP, serial), and MAGE takes advantage of that functionality within a computer network to monitor simulated UAVs.

Because our experimental scenario was conducted locally, and all Flightgear instances and QGroundControl software ran within the same network, it was possible to monitor all simulated UAV locations using a single instance of QGroundControl. In our case, we used the UDP protocol to transfer MAVLink messages between Flightgear and QGroundControl. Each individual instance of Flightgear generates messages for its UAV. Those messages are sent to a specific predetermined IP/port within the network. QGroundControl listens on that IP/port, receives data packets that are generated by each instance of Flightgear, decodes them and displays a position of a UAV in real-time as a pictogram on a 2D horizontal situation map.

Since every Flightgear instance generates messages with a unique ID, QGroundControl can receive and monitor messages from up to 255 UAVs in real time, maintaining their respective positions, attitudes, airspeed, and height. This approach allows for quick and easy assessment of what the group of simulated UAVs is doing in each point in time.

5.3 MAGE Communication Network

The MAGE communication network architecture is developed within Flightgear's Multiplayer system and the *Nasal* environment. As previously indicated, Flightgear's internal capabilities can be extended through Nasal, a scripting language written for the purpose of handling Flightgear's internal properties. *Nasal* stands for 'Not another scripting language' and its scripting concepts are similar to JAVA and Python. It however uses a *tracing garbage collection management system* to automatically determine memory allocation to maximize efficiency. Adjusting Flightgear's many internal properties, such as adding new flight parameters or creating new definitions, can be achieved by scripting them in *Nasal* and loading them into the system when Flightgear is launched.

The communication links within MAGE work upon properties which are transmitted by default over the multiplay system when Flightgear is launched. A *Nasal* file called *Mage.nas* creates a network definition for vehicles in Flightgear. This file is responsible for establishing whether a vehicle connected to the multiplayer server is a part of the game system. When the Flightgear simulator is launched with the file in the *Nasal* directory, a new property node is created in the internal property tree called 'mage'. This node contains two child nodes: current vehicle state information, which contains the state information of the vehicle being controlled, and network vehicle information, which contains details of all other vehicles. The network vehicle information node is a read-only node, i.e. an external system interfacing with the property tree can read this information but cannot manipulate it in any way. The current vehicle contains the state information, instrumentation information and its position in the network topology. Vehicles entering the game system must have the string 'MAGE' appended to the beginning of their callsign to connect to the game system. When a vehicle connects to the system, the state information of the vehicle can be viewed by other vehicles through the *players* node under the *mage* node. The network information system can be written from scratch in *Nasal* to create a network topology between the unmanned vehicles and the manned

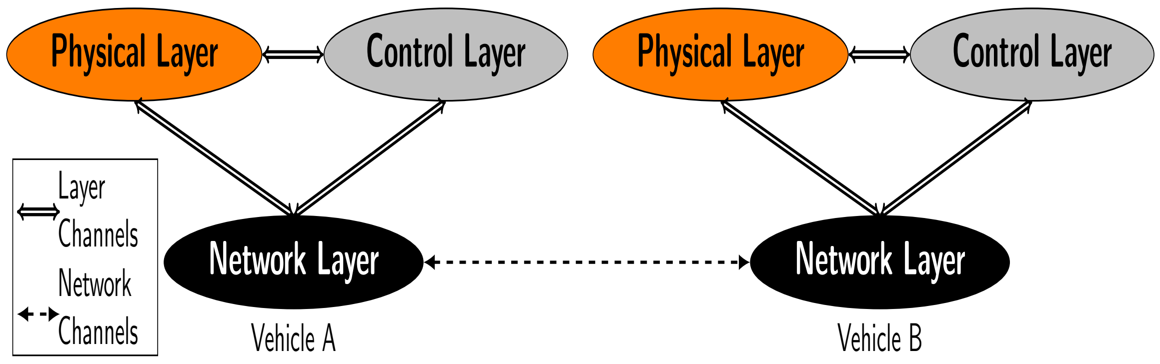


Figure 5.5: Component-Relation diagram of various components of MAGE

vehicles.

5.4 System Setup and Execution

In this section we describe in detail the MAGE system architecture. As previously discussed, the system components are modularized and interact with each other using a UDP socket protocol. The system can be launched either on one or on multiple computers connected to a network. The system consists of three layers: physical layer, network layer, and control layer that communicate with each other using UDP sockets. The architecture is shown in Figure 5.5.

5.4.1 System layers

The base layer is the physical layer which simulates the dynamic model and updates the state information of the vehicle. This layer is derived directly from Flightgear's simulator structure and, once launched, opens up Flightgear's internal property tree which exposes the vehicle's parameters. The state parameters: position, attitude, velocities and accelerations are output using a generic communication protocol on a system generated port.

The next layer is the network layer, which is initialized by *Mage.nas* when Flightgear is launched. The network layer interacts with the physical layer's state information and constructs message packets regarding the state information and then transmits them based on the underlying network protocol. This layer ensures linkage between other network layers of other workstations running the MAGE game system.

The third and final layer is the control layer, which encompasses the control and decision making subcomponents. This layer is a switch-layer, that is the information in this layer can either come from an autonomous system or a human operator. The control layer works by extracting information from the network layer and the physical layer. Thus the data coming into the control layer consists of current vehicle state and neighboring vehicles' messages. In case a human player is operating a vehicle, the control layer offers functionality to connect the system to a human-operated controller. Autonomous control and decision making systems interface with the control layer via UDP and hence the controller can be written in any language with a networking interface capability.

5.4.2 MAGE System Run-Time Execution

The MAGE system is currently being executed from the command line by entering individual parameters with limited options. The system can be initialized to launch a team of unmanned vehicles or a human player operating a single vehicle. The MAGE system will generate a network port number which will open UDP socket connections between Flightgear, the controller and the multiplay server. Future work will include a GUI for the launch system. The parameters that need to be set for launching the MAGE game system are:

- Control type i.e. Human or computer
- Total number of autonomous vehicles being launched from the workstation
- Initial position of vehicles
- IP address of local machine
- Flightgear Multiplay server host address
- Flightgear Multiplay server port

Based on the processing capability of a workstation, the MAGE launch system can initialize more than one vehicle on each workstation. The initial position of the vehicles is input as 2D array of size $3 \times n$. Each array contains the Latitude, Longitude and Altitude and n defines the total number of vehicles being launched from a workstation.

The QGroundControl user interface is also launched separately. The current launch system already has capability to ensure that MAVlink communication is established between Flightgear and QGroundControl to visualize the vehicle telemetry information on QGroundControl.

5.5 Experiments with MAGE: Decentralized Inference of Unknown Function

5.5.1 Decentralized Multi-UAS Collaborative Exploration

In this section we present the results from an experiment scenario tested on the MAGE simulator. This exemplary scenario consists of a group of networked UAVs collaboratively exploring an unknown area while avoiding high risk regions. High risk regions include hostile sites on the ground (such as SAM sites) or enemy aerial vehicle patrol locations. The risk is characterized as function $f(\phi, \theta)$ of latitude(ϕ) and longitude(θ) induced by distribution of enemy assets, and is assumed to be modeled using a Gaussian Process [44]. Enemy assets posing risk are detected using a simple radar emulator defined in *Mage.nas*. The operation of the radar emulator is based on Euclidean distance between the UAV and hostile entities operating in the MAGE environment as per rules of operation defined in section 5.4. The UAVs are limited in the amount of fuel they can carry on-board and are hence constrained to operate within small ranges and restricted to sampling f in some small subsets of the domain $\mathcal{X}$.

5.5.2 Experimental Setup and Results

The MAGE simulation environment utilized here is a group of 3 processing workstations connected together on a Local Area Network and each containing libraries for launching the MAGE game system. The workstations use **Intel Core i7-4790S** processors, **16 GB DDR3** RAM and **250 GB** internal storage. The workstations are labeled S0, S1, and S2. Station S0 is connected to a visual display to monitor the telemetry data on QGroundControl. We denote station S0 as the master station as it allows for monitoring the system and S1 and S2 are denoted as slave workstations. The MAGE system is launched on stations S1 and S2 via a Secure Shell (SSH) on the master terminal through a wrapper function containing information for each server's parameters for launching the system. The input to this wrapper function is the total number of autonomous vehicles running on a particular workstation.

In this experiment a total of four instances are launched: two on S0 and one instance on S1 and S2 respectively. The network topology defined for this system is a fully connected network where each vehicle has a connection to every other vehicle in the system. The network topology for the current simulation has a constraint based on the distance between the vehicles. The entire duration of this experiment is set for 1 hour with respect to the master's clock. The master terminates the connections after the period of one hour is over.

During the initial stages of the simulation the UAVs explore randomly and pass the learned model over the communication network to their neighbors. Once a UAV obtains a neighbor's model and fuses it together with its own model, it starts exploring regions that other agents have not visited and are not likely to visit. This is achieved by exploring based on the uncertainty in the Gaussian Process model. Each UAV places a waypoint in an area where the uncertainty in the model it has fused together is highest. This leads to a joint decentralized exploration policy. The effect of the policy is seen in Figure 5.6 where it can be seen that each UAV is heading in different directions so that together they can uncover a larger amount of space over time. The total area uncovered at the end of the simulation is shown in Figure 5.7a and figure indicates that UAVs jointly maximized their exploration.

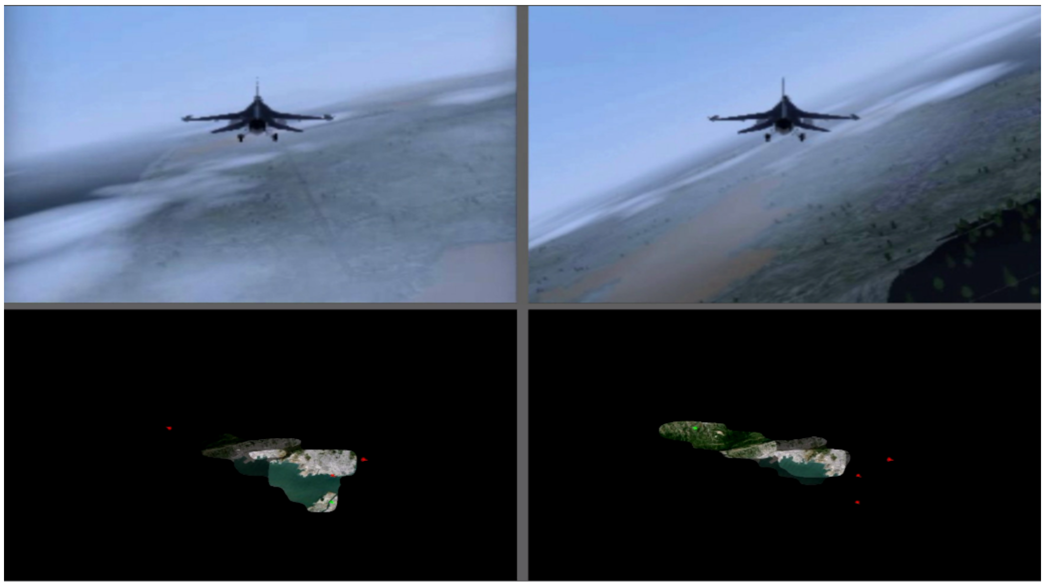
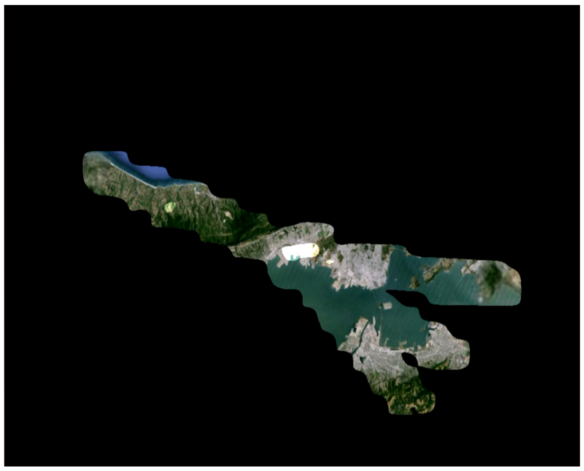
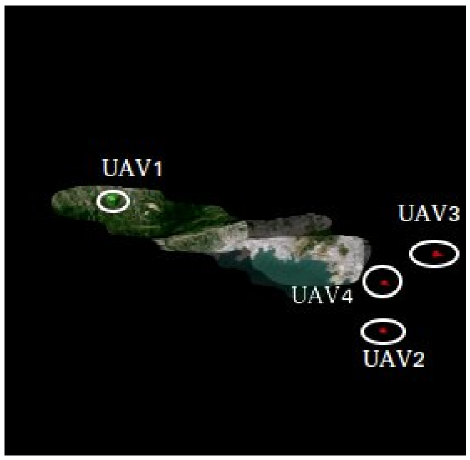


Figure 5.6: Exploration from the perspective of two UAVs, thirty minutes after scenario start. The UAVs have collaborated in decentralized fashion to explore different areas of the scenario space.



(a) Total area uncovered through decentralized inference by four UAVs at the end of simulation



(b) Figure shows four UAVs exploring terrain, from the perspective of UAV1. The other three UAVs are investigating areas which UAV1 has neither visited nor incorporated into its map.

Figure 5.7: Flightgear Multiplayer System

Chapter 6

Conclusions and Future Work

The first part of this thesis introduces a novel algorithm, GP-Fusion for communication efficient decentralized inference using a spatially separated network. The algorithm was empirically demonstrated to have improved communication performance, in comparison with existing state-of-the-art decentralized inference methods. GPFusion and its variation GPFusion-AVoIDs are compared on one synthetic and two real world datasets and are shown is have orders of magnitude improvement in communication while having error performance similar to the current decentralized estimation methods, without having to assume a common set of kernels.

In the second part of this thesis a high-fidelity, multi-agent flight-simulator corroboration of collaborative autonomy algorithms for multi-agent UAS missions is presented. The key distinction of MAGE is its ability to realistically simulate flight-relevant effects, and enable realistic multi-agent interactions where a different computer or human can control each agent. MAGE is developed from existing open source software and is modularized for efficient extensions for many possible multiagent scenarios. A communication network was developed by manipulating the internal properties of the Flightgear flight simulator, allowing the efficient definition of a graphical network topology for passing information between agents and humans.

One potential direction for extending GP-Fusion lies in designing methods which can track the temporal evolution of the underlying process. A set of methods which deal with non-staionary kernels are being discussed in literature and hence potentially it will be interesting to study how fusion can be achieved with time varying kernels.

Bibliography

[1] Nisar Ahmed, Jonathan Schoenberg, and Mark Campbell. Fast weighted exponential product rules for robust general multi-robot data fusion. In *Robotics: Science and Systems (RSS)*, Sydney, Australia, July 2012.

[2] Rakshit Allamaraju, Hassan Kingravi, Allan Axelrod, Girish Chowdhary, Robert Grande, Christopher Crick, Weihua Sheng, and Jonathan How. Human aware path planning in urban environments with nonstationary mdps. In *International Conference on Robotics and Automation*, Hong Kong, China, 2014. IEEE.

[3] Mauricio A Alvarez, Lorenzo Rosasco, and Neil D Lawrence. Kernels for vector-valued functions: A review. *arXiv preprint arXiv:1106.6251*, 2011.

[4] Christopher Amato, Daniel S. Bernstein, and Shlomo Zilberstein. Solving POMDPs using quadratically constrained linear programs. In *International Joint Conference on Artificial Intelligence*, 2007.

[5] Christopher Amato, Girish Chowdhary, Alborz Geramifard, N Kemal Ure, and Mykel J Kochenderfer. Decentralized control of partially observable markov decision processes. In *The 52nd IEEE Conference on Decision and Control (CDC)*, 2013.

[6] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[7] Okan Asik and H Levent Akin. Solving multi-agent decision problems modeled as dec-pomdp: A robot soccer case study. In *RoboCup 2012: Robot Soccer World Cup XVI*, pages 130–140. Springer, 2013.

[8] Saptarshi Bandyopadhyay and Soon-Jo Chung. Distributed estimation using bayesian consensus filtering. In *American Control Conference (ACC), 2014*, pages 634–641. IEEE, 2014.

[9] Michael Basler, Martin Spott, Stuart Buchanan, Jon Berndt, Bernhard Buckel, Cameron Moore, Curt Olson, Dave Perry, Michael Selig, Darrell Walisser, et al. The flightgear manual. *The FlightGear Manual*, 2010.

[10] Frederic Bourgault, Tomonari Furukawa, and Hugh F Durrant-Whyte. Decentralized bayesian negotiation for cooperative search. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2681–2686. IEEE, 2004.

[11] Jie Chen, Kian Hsiang Low, and Colin Keng-Yan Tan. Gaussian process-based decentralized data fusion and active sensing for mobility-on-demand system. *arXiv preprint arXiv:1306.1491*, 2013.

[12] Chee-Yee Chong and Shozo Mori. Convex combination and covariance intersection algorithms in distributed fusion. In *Proc. of the 4th International Conference on Information Fusion*, 2001.

[13] Girish Chowdhary, Jonathan P. How, and Hassan Kingravi. Bayesian nonparameteric adaptive control using gaussian process. In *Neural Information and Processing Systems*, Lake Tahoe, NV, December 2012. workshop on Bayesian Nonparametric Models For Reliable Planning And Decision-Making Under Uncertainty.

[14] Noel Cressie and Christopher K Wikle. *Statistics for spatio-temporal data*. John Wiley & Sons, 2011.

[15] Lehel Csatö and Manfred Opper. Sparse on-line gaussian processes. *Neural computation*, 14(3):641–668, 2002.

[16] H. F. Durrant-Whyte, S. Majumder, M. de Battista, and S. Scheding. A Bayesian algorithm for simultaneous localisation and map building. In R. Jarvis and A. Zelinsky, editors, *Robotics Research: The Tenth International Symposium*, Victoria, Australia, 2001.

[17] Hugh Durrant-Whyte, Mike Stevens, and E Nettleton. Data fusion in decentralised sensing networks. In *4th International Conference on Information Fusion*, pages 302–307, 2001.

[18] Staffan Ekvall and Danica Kragic. Robot learning from demonstration: a task-level planning approach. *International Journal of Advanced Robotic Systems*, 5(3):223–234, 2008.

[19] Antonio Franchi, Cristian Secchi, Markus Ryll, Heinrich H Bülthoff, and Paolo Robuffo Giordano. Shared control: Balancing autonomy and human assistance with a group of quadrotor uavs. *Robotics & Automation Magazine, IEEE*, 19(3):57–68, 2012.

[20] Dietrich Franken and Andreas Hüpper. Improved fast covariance intersection for distributed data fusion. In *Information Fusion, 2005 8th International Conference on*, volume 1, pages 7–pp. IEEE, 2005.

[21] Robert C. Grande. Computationally efficient Gaussian process changepoint detection and regression. Master's thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA, June 2014.

[22] Qing Guo, Siyue Chen, Henry Leung, and Shutian Liu. Covariance intersection based image fusion technique with application to pansharpening in remote sensing. *Information Sciences*, 180(18):3434–3443, 2010.

[23] Simon J Julier and Jeffrey K Uhlmann. Using covariance intersection for slam. *Robotics and Autonomous Systems*, 55(1):3–20, 2007.

[24] Ahmed Tashrif Kamal, Chong Ding, Bi Song, Jay A Farrell, and AK Roy-Chowdhury. A generalized kalman consensus filter for wide-area video networks. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, pages 7863–7869. IEEE, 2011.

[25] Hassan A. Kingravi. Approximation algorithms for vector-valued rkhss. Technical report, Georgia Institute of Technology, 2014.

[26] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research (JMLR)*, 9:235–284, June 2008.

[27] Chris M. Kreucher, Alfred O. Hero, and Keith D. Kastella. An information-based approach to sensor management in large dynamic networks. *Proc. IEEE, Special Issue on Modeling, Identificiation,& Control of Large-Scale Dynamical Systems*, 95(5):978–999, May 2007.

[28] W. Liu, J. C. Principe, and S. Haykin. *Kernel Adaptive Filtering: A Comprehensive Introduction.* Wiley, Hoboken, New Jersey, 2010.

[29] Kevin M Lynch, Ira B Schwartz, Peng Yang, and Randy A Freeman. Decentralized environmental modeling by mobile sensor networks. *Robotics, IEEE Transactions on*, 24(3):710–724, 2008.

[30] A Makarenko, A Brooks, T. Kaupp, H. Durrant-Whyte, and F. Dellaert. Decentralised data fusion: A graphical model approach. In *Information Fusion, 2009. FUSION '09. 12th International Conference on*, pages 545–554, July 2009.

[31] Alexei Makarenko and Hugh Durrant-Whyte. Decentralized data fusion and control in active sensor networks. In *Proceedings of the Seventh International Conference on Information Fusion*, volume 1, pages 479–486, 2004.

[32] Lorenz Meier, JF Camacho, B Godbolt, J Goppert, L Heng, M Lizarraga, et al. Mavlink: Micro air vehicle communication protocol. *Online]. Tillgänglig: http://qgroundcontrol.org/mavlink/start.[Hämtad 2014-05-22]*, 2013.

[33] Mehran Mesbahi and Magnus Egerstedt. *Graph theoretic methods in multiagent networks*. Princeton University Press, 2010.

[34] Randolph L Moses, Martin J Wainwright, and Alan S Willsky. Distributed fusion in sensor networks. *IEEE Signal Processing Magazine*, pages 42–55, 2006.

[35] Eric J Msechu and Georgios B Giannakis. Distributed measurement censoring for estimation with wireless sensor networks. In *Signal Processing Advances in Wireless Communications (SPAWC), 2011 IEEE 12th International Workshop on*, pages 176–180. IEEE, 2011.

[36] Beipeng Mu, Girish Chowdhary, and Jonathan P. How. Efficient distributed sensing using adaptive censoring-based inference. *Automatica*, 50(6):1590 – 1602, 2014.

[37] R. Olfati-Saber. Distributed Kalman filtering for sensor networks. In *Decision and Control, 2007 46th IEEE Conference on*, pages 5492–5498. IEEE, 2007.

[38] Reza Olfati-Saber, J Alex Fax, and Richard M Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[39] Jos GJ Olivier, John A Van Aardenne, Frank J Dentener, Valerio Pagliari, Laurens N Ganzeveld, and Jeroen AHW Peters. Recent trends in global greenhouse gas emissions: regional trends 1970–2000 and spatial distributionof key sources in 2000. *Environmental Sciences*, 2(2-3):81–99, 2005.

[40] Curtis L. Olson. Simgear - simulator construction tools, 2010.

[41] Lee-Ling Ong, Tim Bailey, Hugh Durrant-Whyte, and Ben Upcroft. Decentralised particle filtering for multiple target tracking in wireless sensor networks. In *Information Fusion, 2008 11th International Conference on*, pages 1–8. IEEE, 2008.

[42] Alexander R Perry. The flightgear flight simulator. In *Proceedings of the USENIX Annual Technical Conference*, 2004.

[43] BSY Rao, HF Durrant-Whyte, and JA Sheen. A fully decentralized multi-sensor system for tracking and surveillance. *The International Journal of Robotics Research*, 12(1):20–44, 1993.

[44] Carl Edward Rasmussen, Malte Kuss, et al. Gaussian processes in reinforcement learning. In *NIPS*, volume 4, page 1, 2003.

[45] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005.

[46] Wei Ren, Randal W Beard, and Ella M Atkins. A survey of consensus problems in multi-agent coordination. In *American Control Conference, 2005. Proceedings of the 2005*, pages 1859–1864. IEEE, 2005.

[47] Bernhard Scholkopf, Ralf Herbrich, and Alex Smola. A generalized representer theorem. In David Helmbold and Bob Williamson, editors, *Computational Learning Theory*, volume 2111 of *Lecture Notes in Computer Science*, pages 416–426. Springer Berlin / Heidelberg, 2001.

[48] B. Scholk?pf and A. Smola. *Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press, Cambridge, MA, USA, 2002.

[49] Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer, 1999.

[50] Leila Takayama. Telepresence and apparent agency in human–robot interaction. *The Handbook of the Psychology of Communication Technology*, pages 160–175, 2015.

[51] Wee Peng Tay, John N Tsitsiklis, and Moe Z Win. Asymptotic performance of a censoring sensor network. *Information Theory, IEEE Transactions on*, 53(11):4191–4209, 2007.

APPENDIX A

Acronyms

Acronym	Expanded Version
MEMS	MicroElectro Mechanical System
ASN	Autonomous Sensor Networks
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
GPF	Gaussian Processes Fusion
GP	Gaussian Processes
BNP	Bayesian Non-Parametric
DDF	Decentralized Data Fusion
GKCF	Generalized Kalman Consensus Filter
CI	Covariance Intersection
CF	Channel Filters
VoI	Value of Information
KLD	Kulbeck-Leibler Divergence
AVoIDs	Adaptive Value of Information Distributed Sensing
RKHS	Reproducing Kernel Hilbert Spaces

VITA

Rakshit D. Allamraju

Candidate for the Degree of

MASTER OF SCIENCE

Thesis: COMMUNICATION EFFICIENT DECENTRALIZED INFORMATION FUSION IN SENSOR NETWORKS

Major Field: Mechanical And Aerospace Engineering

Biographical:

Education: Completed the requirements for the Master's of Science degree with a major in Mechanical and Aerospace Engineering at Oklahoma State University in December, 2015. Completed requirements for Bachelors of Technology degree with a major in Avionics Engineering at University of Petroleum and Energy Studies in May, 2012.

Experience:

Research Assistant for DasLab, at Oklahoma State University, Stillwater, OK: Summer 2013 to present.

Teaching Assistant for Senior Design Projects (MAE 4344), at Oklahoma State University, Stillwater, OK: Spring 2015

Professional Memberships: AIAA and IEEE.