University of Colorado, Boulder
CU Scholar

Aerospace Engineering Sciences Graduate Theses & Dissertations

Aerospace Engineering Sciences

Spring 1-1-2017

Cooperative Robot Localization Using Event-Triggered Estimation

David I. Iglesias Echevarria
University of Colorado at Boulder, iglesias.david.ignacio@gmail.com

Follow this and additional works at: https://scholar.colorado.edu/asen_gradetds

Part of the Aerospace Engineering Commons, and the Robotics Commons

Recommended Citation
Iglesias Echevarria, David I., "Cooperative Robot Localization Using Event-Triggered Estimation" (2017). *Aerospace Engineering Sciences Graduate Theses & Dissertations*. 206.
https://scholar.colorado.edu/asen_gradetds/206

This Thesis is brought to you for free and open access by Aerospace Engineering Sciences at CU Scholar. It has been accepted for inclusion in Aerospace Engineering Sciences Graduate Theses & Dissertations by an authorized administrator of CU Scholar. For more information, please contact cuscholaradmin@colorado.edu.

www.manaraa.com

Cooperative Robot Localization Using Event-Triggered Estimation

by

David I. Iglesias Echevarria

B.S., Polytechnic University of Catalonia, 2015

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Aerospace Engineering Sciences

2017

www.manaraa.com

This thesis entitled:
Cooperative Robot Localization Using Event-Triggered Estimation
written by David I. Iglesias Echevarria
has been approved for the Department of Aerospace Engineering Sciences

_____

Prof. Nisar R. Ahmed

_____

Prof. Eric W. Frew

_____

Prof. Jay W. McMahon

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

المنارة للاستشارات
www.manaraa.com

Iglesias Echevarria, David I. (M.S., Aerospace Engineering)

Cooperative Robot Localization Using Event-Triggered Estimation

Thesis directed by Prof. Nisar R. Ahmed

It is known that multiple robot systems that need to cooperate to perform certain activities or tasks incur in high energy costs that hinder their autonomous functioning and limit the benefits provided to humans by these kinds of platforms. This work presents a communications-based method for cooperative robot localization. Implementing concepts from event-triggered estimation, used with success in the field of wireless sensor networks but rarely to do robot localization, agents are able to only send measurements to their neighbors when the expected novelty in this information is high. Since all agents know the condition that triggers a measurement to be sent or not, the lack of a measurement is therefore informative and fused into state estimates. In the case agents do not receive either direct nor indirect measurements of all others, the agents employ a covariance intersection fusion rule in order to keep the local covariance error metric bounded. A comprehensive analysis of the proposed algorithm and its estimation performance in a variety of scenarios is performed, and the algorithm is compared to similar cooperative localization approaches. Extensive simulations are performed that illustrate the effectiveness of this method.

www.manaraa.com

Dedication

To my parents, for helping me be who I am today. To my partner, for lighting the way when it was dark, and our love-hate relationship for pizzas.

This thesis is more yours than mine.

Proposition 1:

Somewhere, something incredible is waiting to be known.

Carl Sagan

Proposition 2:

This is all very confusing.

Student in the Aerospace Grad Lounge

Corollary:

Even if things get tough we will keep pushing, for the reward is worth it.

www.manaraa.com

Acknowledgements

I want to express my deepest gratitude, first and foremost, to my advisor, Prof. Nisar Ahmed. He has not only proved to be an endless source of technical knowledge, but has also offered, in countless meetings and conversations, the kind of guidance and wisdom that deeply impact one's personal development. I will always be grateful to him for the many opportunities he has given me during my time as a graduate student.

Secondly, I want to thank all the professors that I have had here at the University of Colorado at Boulder for providing a first-class education that is both thorough and deeply humane. A special thanks to the members of my committee, for kindly accepting to review this work and for their support.

Finally, I want to wholeheartedly thank the Balsells Foundation, the Generalitat de Catalunya and the University of Colorado at Boulder for the once-in-a-lifetime opportunity to pursue graduate studies in the United States through the Balsells Fellowship Program. A lot of people are involved in this program and go out of their way to make sure that everything works smoothly year after year, so it would be impossible to thank all of them individually. However, I can not continue without mentioning a few of them. First, Mr. Pere Balsells, the forefather of this truly amazing community, for a life of giving and generosity that inspires me beyond words; Prof. Robert Davis, for taking care of each one of us even while he was appointed as Dean of the College of Engineering and Applied Science; and finally, Ms. Sharon Powers, former Dean's Assistant, for making this place home for us, even when home is thousands of miles away.

www.manaraa.com

Contents

Chapter

www.manaraa.com

www.manaraa.com

Tables

Table

www.manaraa.com

Figures

Figure

www.manaraa.com

www.manaraa.com

www.manaraa.com

Chapter 1

Introduction

1.1 The quest for autonomy

In recent years, there has been an increasing number of research activities in the subject of robotic vehicle networks, propelled by the miniaturization of electromechanical components (sensor suites, actuators, etc.), lower acquisition costs and improvement of wireless ad hoc networks. Nowadays, mobile robots can be found in a myriad of environments and sectors, ranging from domestic services (e.g. cleaning robots), intelligent transportation and construction and environmental monitoring, to scientific exploration (e.g. the different Mars rovers) and military (e.g. surveillance drones). This vast set of applications imposes that there is not one ideal platform to perform them all – whereas in some cases a unique, more powerful, well-equipped and expensive robot may be desirable, in others it may be more suited to deploy a fleet of less advanced units that, on the other hand, are able to sense, communicate and cooperate with one another.

This interest has been not just accompanied, but also fostered by crucial advances in the field of artificial intelligence and its most immediate effect on robot systems, *autonomy*. What we define by autonomy is somewhat elusive and abstract at a technical level, although a practical, high-level way to describe it is as the ability of a system to make its own decisions, without any need for human input or guidance in the process. This is of great interest by both academia and industry, and from both a theoretical and practical standpoint, since increasingly autonomous systems have a potentially huge effect on our economy, society and the way we live in general. Autonomous robot systems can not only outperform humans at some tasks, but also enable us to pursue entirely new

www.manaraa.com

endeavors.

1.2 The importance of robot localization

Precise localization is one of the main requirements for autonomy in mobile robot autonomous systems. Many of the tasks these autonomous robots are to perform rely on accurate knowledge of their position and orientation in space. Localization, in the context of mobile robots, can be seen as a problem of coordinate transformation between the robot's reference frame and the world's reference frame. Traditionally, robots use a combination of *proprioceptive* and *exteroceptive* measurements to compute estimates of their poses (composed by both position and orientation). The former refers to measurements taken to the measuring robot only (e.g. those obtained from the wheel encoder or the inertial measurement unit), while the latter refers to measurements taken between the measuring robot and the environment (e.g. relative range to a landmark, environment perception through camera), or between the measuring robot and other robots in the network (e.g. relative range and bearing between robots).

Although for some applications robots can benefit from access to GPS for accurate absolute positioning, in many others, such as in indoor environments and planetary exploration missions, they can not. We can consider yet a third scenario, in which robots can only sporadically get absolute position information, subsequently having to perform their goal tasks relying on this low-frequency, scarce knowledge combined with higher-frequency relative measurements and odometry. An example of this are underwater robots, which have to resurface often to get a GPS fix for their estimates, due to the inherent limitations of dead reckoning. This poses the problem of how to have a system that remains operational and useful under these challenging circumstances.

There is a variety of ways by which robot systems are able to localize the agents in the network. Most methods in the last couple of decades, including this work, have followed what we call a *probabilistic* view of the field. Probabilistic robotics is the study of robots and their interactions with the environment by understanding the role and modeling the effects that the inherent uncertainties that robots encounter play. Traditional approaches within this subfield

www.manaraa.com

consider that the robots are equipped with sensors of different kinds with which they perceive the world around them and find *estimates* of their position in it.

While that is perfectly valid, a new conception of localization for multiple-robot systems emerged with strength a few years ago. This strategy to deal with the multi-robot localization problem is what in the literature is referred to as *cooperative localization* (CL), and will be discussed extensively in following sections since it is the minimal problem that encompasses this thesis. The main idea behind CL is to use relative measurements from vehicle to vehicle to *jointly* estimate the poses of all team members, which results in increased accuracy for all agents [2]. The principle behind CL is to exploit correlations existing between the robots' states (in this case, the poses) by taking and fusing relative measurements. Because of this coupling between states, sporadic access to accurate position/orientation information by one agent directly benefits the estimate for the entire network, thanks to the ability to pin down that particular agent.

1.3 Towards increased battery life

In a robotic system, there are many activities that the vehicles carry out and that require energy; moving around, communications, processing and storing data are only a few examples. Being able to accurately localizing a team of mobile robots is different to doing so *efficiently*. Here, the word efficient is used in the most generic sense, meaning that the system incurs in little energy costs. As intuition suggests, efficient systems translate to longer deployment times, less demanding sensor, bandwidth and processing requirements and, in turn, less costly hardware, thus making the robot platform more versatile. In some cases, the difference between having an algorithm that handles data more efficiently than another is the ability to adequately perform the mission itself, as seen, for example, in [3]. While advances in battery technology and embedded systems are promising and have already had a vast effect in the way we imagine and use robots, having a limited usage time will always be a hurdle to overcome, thus making the study of algorithms a topic of perpetual interest.

Robot systems are comprised of, put very simply, two main elements that interact with

www.manaraa.com

each other: hardware and software. The design of a system for a specific application will involve a back and forth process in which some hardware components will be fixed, the software will be incorporated, the system will be tested and, based on the performance and requirements of the problem, the process will be repeated, hopefully with less unknowns. As one can expect, the differences concerning power, processor speed, sensor quality, etc., between working with a platform that is equipped with state-of-the-art components and one that is inexpensive, light and commercially accessible are not minor. In spite of that, a behemoth robot is rarely the solution in mobile robotics, for obvious reasons. For example, in aerospace applications lightweight vehicles are preferred since weight is a severely limiting factor when it comes to things that fly. The specifications of the hardware will therefore play a crucial role on the capabilities of our system. Within the boundaries and stringent conditions set by hardware, however, software will need to be implemented in such a way that our robots are as useful and able to perform the task at hand as possible. Several approaches have been developed to further push the edge of what is possible for robotic systems that are constrained by the components they use or by the application they are designed for.

In this work, a communications-based method is developed and studied. The method proposed stems from the fact that in cooperative localization, a lot of data is sent and received by the agents. While this is beneficial for localization purposes, it brings about communication costs that do not scale well with the network size. With the goal of reducing these costs, the approach taken builds on the idea that the value of observations agents take can somehow be measured. In particular, agents can decide that if some piece of information is not useful enough, as determined by a threshold, it will not be sent. The agent that is meant to receive that information then uses its knowledge of this threshold to infer *something* about the original data, thus still benefiting from the lack of explicit information. This notion of sending data whenever a condition is met or an event takes place is known in the literature as *event-based* estimation, and allows for significant energy savings and can reduce communication bandwidth in networked control systems and wireless sensor networks [4].

www.manaraa.com

As stated, this is one of many way that pushes us forward to fully autonomous systems. Robotics and autonomy being two of the topics that are receiving most attention and scrutiny worldwide, the hope is that this work will be of service for future endeavors.

1.4 Thesis goals and overview

This work is born with the main objective of understanding cooperative localization a little better by answering perhaps the most fundamental question to be solved, which is how can we perform cooperative localization in an accurate, consistent, scalable, robust and energy-efficient way? In order to do that, many subquestions need to be posed and answered as well, for which a good grasp of current methods and research tendencies is necessary. Given the finding that in wireless ad hoc networks a lot of energy is spent on data sharing, this work focuses on a solution that deals with the problem of making a better use of that data. With that in mind, we attempt to address issues such as the sensitivity of estimation performance to the measurement sharing rate, to data packets losses, to network topology, or to sensor types; the algorithm failure modes; the range of safe applicability of the filter, given that the system is nonlinear and there are no guarantees that it will work properly; and the performance of our proposed method when compared to a baseline method, among others.

This thesis is organized as follows: Chapter 2 gives a short summary of basic concepts and outlines some well known algorithms in estimation. Additionally, we perform a review of related work on cooperative robot localization. Chapter 3 formally introduces the problem of cooperative localization, which is the central topic of this work. Different approaches to the problem are presented, together with their strengths and shortcomings, in order to help the reader build a mental map of the state of the field. In Chapter 4, our proposed solution to the cooperative localization problem is examined in detail, trying to justify its relevance and interest. Chapter 5 provides a detailed look at the computer simulations that have been performed to better understand the performance of our algorithm, since they play a crucial role. Continuing with that, in Chapter 6 the results of the simulations are presented and discussed. Finally, Chapter 7 is comprised of the

www.manaraa.com

conclusions from all experiments and discussions, and a section with recommendations for future work in the topic.

www.manaraa.com

Chapter 2

Background and related work

In this section, we introduce a brief review of basic concepts in state estimation and we present related work other research groups and institutions have done in robot localization. The basic concepts of probability are not developed here for the sake of brevity, but they can be found in any introductory probability textbook. The reader that is familiar with the basics of estimation theory can safely skip Section 2.1 and go directly to Section 2.2.

2.1 Concepts in estimation

In state estimation, we use the laws of statistics and probability to develop frameworks that allow us to infer quantities that are not directly observable from noisy data. In particular, in robot localization we usually care about the position and orientation of the robots with respect to their environment. Probabilistic methods have proven really useful, since with them we can model the different uncertainties that exist in our systems.

One of the most popular approaches to state estimation in robotics is the Bayes filter. The Bayes filter is, essentially, an algorithm that uses Bayes' theorem to develop an expression to recursively compute the state's probability density function taking into account observations and the system's properties, such as how (if it does) the state evolves with time and to inputs.

The Bayes filter is a theoretical model, a framework that allows us to derive the fundamental expressions to do recursive estimation. In reality, we need ways to describe and compute numerically the probability distributions that appear in the Bayes filter equations, in a lot of cases having to

www.manaraa.com

make assumptions to make our problem more tractable. There are different approaches to do this. One of them are Gaussian filters, which represent the probability distributions with normal distributions described by two parameters, mean and covariance. This type of filters is useful in a lot of practical situations, but the properties of normal distributions limit the applicability of Gaussian filters. One issue is that normal distributions are unimodal, that is, they have one (and only one) maximum. As an example of a situation that is not apt for a Gaussian filter, let us imagine that a robotic vehicle is moving along a road and it encounters a fork. At this point, we most likely need to model the state's posterior with a multimodal distribution, since at least at the beginning, there is some uncertainty as to which one of the paths the vehicle is on. Despite their shortcomings, Gaussian filters are successfully used in a variety of tracking applications in which the posterior is centered around the true state with some uncertainty.

The Kalman filter (KF) is perhaps the most widely used Gaussian filter, and its origin can be traced back to both [5] and [6]. The KF is used for continuous state spaces, and assumes that the probability that models how states evolve in time given some dynamics and control inputs (called state transition probability) is a linear function with added Gaussian noise. It also assumes that the probability that models how a state affects an observation (called observation or measurement probability) is a linear function with Gaussian noise added to it. If the system verifies these conditions, then the Kalman filter is the optimal minimum mean squared error (MMSE) estimator. However, because of the assumptions of linearity in both the dynamics and observations, this approach is severely limited, given that very few things in real life are truly linear. This raises the next question: how can we perform estimation in nonlinear systems?

Multiple approaches have been developed to do nonlinear estimation. One of these approaches is what we call the extended Kalman filter (EKF). The EKF acknowledges that the transition and measurement probabilities will usually not be linear functions of the state, and relaxes this assumption. In an EKF, both these probabilities are nonlinear functions. However, the equations derived for the KF no longer hold without the linearity assumption. In order to deal with that, the EKF *approximates* the true nonlinear model with a linear one by linearizing these functions

www.manaraa.com

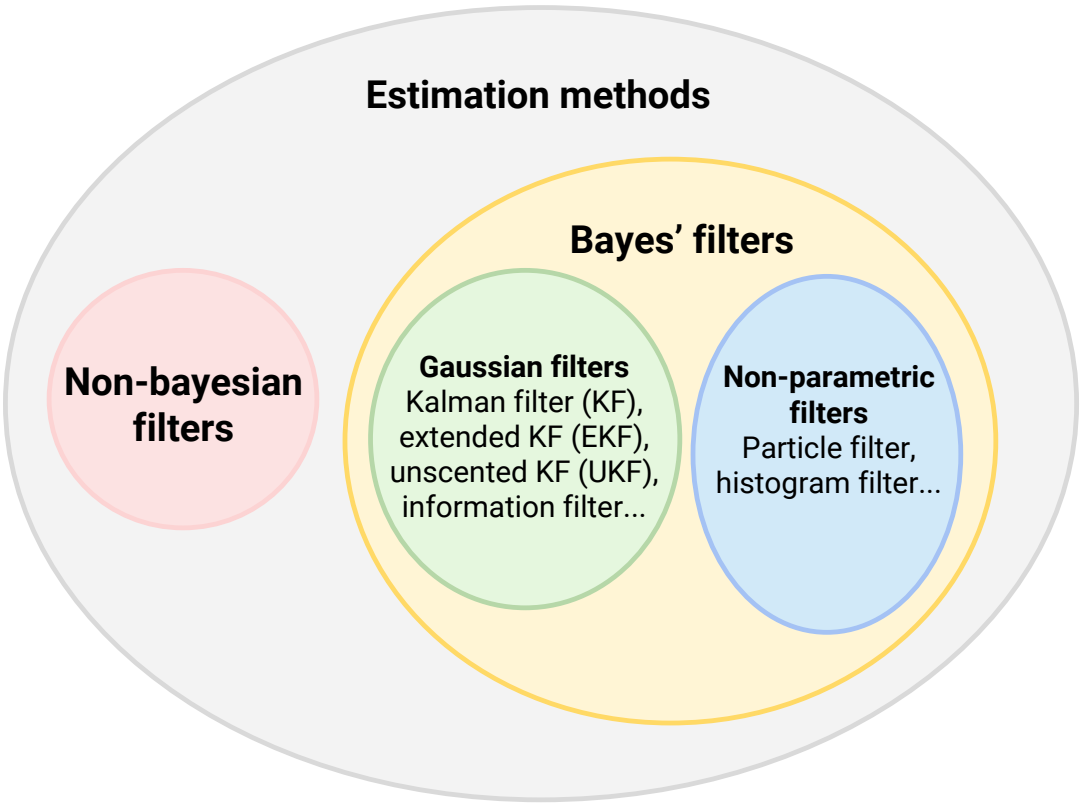


Figure 2.1: Diagram showing a classification of some of the most known estimation algorithms.

www.manaraa.com

around a particular point. This implies that, unlike the KF, the EKF is an approximate solution and does not offer any guarantees that the output will be correct.

The performance of the EKF depends chiefly on two aspects: the degree of nonlinearity around the linearization point of the approximated functions, and the degree of uncertainty in our estimates. The first one is due to the fact that we are approximating mappings with linear transformations; the more nonlinear the original function is, the less accurate our representation will be. The second one stems from the first one. The higher the uncertainty in the estimate, the wider its probability density function will be, thus being more distant from the point of linearization and more affected by the transformation. Despite being an approximate solution, the EKF works remarkably well in a lot of scenarios in which the nonlinear effects are not very pronounced, and has been in fact used repeatedly in the robotics industry due to the relative simplicity of the algorithm, both conceptually and computationally. The present work uses an EKF framework all throughout for the already mentioned reasons.

2.2 Related work

2.2.1 Cooperative localization

The idea of exploiting vehicle-to-vehicle measurements to perform robot localization can be traced back to [7] and its follow-up works, [8, 9]. In them, the authors propose a method for localization in systems with multiple robots that results in lower positioning error than dead reckoning when no landmarks are available. They achieve that by dividing the team into two groups that take turns while moving in an environment until they reach the destination point. In this process, the group that remains stopped acts as a landmark for the other team.

Since then, cooperative localization has received extensive interest from different communities, including those of robotics and wireless communications. A few years after that first work, one of the first probabilistic approaches using a Markov localization algorithm (which can be found in [10, 11, 12, 13]) was detailed in [14], where robots use odometry and environment measurements

www.manaraa.com

to update their own local belief function when they can not detect other robots. When two robots are close enough to detect each other, they take relative measurements that are transformed into density trees that are used to refine the estimates. A considerable amount of relatively similar work along those lines has been developed more recently, where agents use of a combination of proprioceptive and exteroceptive measurements to compute their localization. In [15], the authors describe what became one of the first approaches to CL that is within a Kalman filter framework. The paper considers a group of vehicles that move in an environment and that can sense one another. The algorithm is also decentralized, hence being more rebust and scaling better than its centralized equivalents, and each vehicle only keeps an estimate of its own state and uses odometric measurements to compute its localization. Only when two agents are at a sufficiently small distance do they measure each other and exchange this information. Their results show a significant improvement in position accuracy when relative measurements used, even if only intermitently. One of the limitations of this latter paper, however, is that vehicles use relative pose measurements, which are not common in many real-life situations. Within the last half decade, [16] has proposed a similar decentralized CL approach that uses covariance intersection (CI) as its main fusion method instead of a KF. Their proposed method aims at reducing communication and processing complexity in large networks of robots, and is provably consistent (see [17]), even though decentralized CI has the disadvantage of giving suboptimal results (see [18]). Additionally, this paper uses a model for the observations that is not commonplace, its interest hence being more limited considering our purposes. Breaking a lot of these assumptions we can find [19], where the observations are nonlinear, data transmission is asynchronous and lossy and the initial poses are not known. The authors use a particle filter to compute the estimates. A shortcoming that is shared by all these works is the fact that, even though they study multi-agent systems, the number of vehicles always stays small, thus not considering the effects that different communication strategies have on the performance of the algorithm.

www.manaraa.com

2.2.2 Event-based estimation

The idea of event-based estimation is newer than that of cooperative localization and, to the best of our knowledge, all the references date from the last 10 years. The main idea of event-based algorithms is to limit the amount of data transmitted in wireless sensor networks (WSNs) by just sharing information when an *event* (for example, a certain signal exceeding a predetermined threshold) occurs. In [20], the authors propose a general centralized state estimation algorithm that can process event-based measurements and that is based on a Gaussian mixture model (GMM) to make it more computationally tractable. Results show that, using a hybrid strategy where both received and not received measurement data is used for update through different algorithms, the covariance matrix remains bounded even when no measurements are received anymore. A similar algorithm can be found in [21], where a Kalman filtering framework is used instead of a GMM. [22] characterized the effects of multiple sensors with each sensor channel having its own event-triggering condition on the MMSE estimates. In particular, the paper shows that the estimates are determined by the conditional mean and covariance of the innovations. However, these results are valid for the case with linear processes and observations.

While most previous endeavors have been focused on finding analytic solutions to the problem of event-triggered measurement fusion in WSNs, thus assuming simplified system models, we feel that not enough work has been done on studying the implementation and performance of these algorithms to more realistic scenarios. Our work tries to answer some of the natural questions that follow from an engineering perspective by proposing an approximate method and relaxing many of the previous assumptions.

www.manaraa.com

Chapter 3

The cooperative localization problem

In general terms, cooperative localization is the problem of determining the poses (position and orientation) in a given environment of a set of points based on a series of relative measurements between these points. For us, these points are robots (also called agents) that have both communication and sensing capabilities, and that move in the environment, so the problem transforms into mobile cooperative localization. The robots measure one another and share these measurements either with other agents or with a fusion center that does all the data processing, so that we are able to localize them. Mobile cooperative localization is an instance of the general localization problem, which is one of the fundamental problems in robotic perception [23].

As we have seen, robots are equipped with sensors that they use to measure themselves (what we call *proprioceptive* measurements) or to measure other vehicles or the world around them (what we call *exteroceptive* measurements). The main difficulty with robot localization is that the thing that we are trying to determine, in this case the poses of the agents in our system, can generally not be measured directly by the sensors. On top of not measuring directly the quantities that we are interested in knowing, the data points provided by sensors are noisy. These two issues make it necessary, in the vast majority of cases, to use numerous data points, and in fact, the more data points we use to estimate the desired quantity, the more confident we will be in our results. This is where probabilistic filters, such as the Kalman filter, prove to be so useful.

www.manaraa.com

3.1 Notation

Here, we introduce some useful notation and concepts from probability theory. Let $\mathbb{R}$ be the set of real numbers, and $\mathbb{R}_{\geq 0}$ the set of real numbers that are zero or positive. For a vector $v \in \mathbb{R}^d$, let $\text{diag}(v) \in \mathbb{R}^{d \times d}$ the diagonal matrix made with the values of v. Both $\texttt{elem}(v, k)$ and $(v)_k$ designate the k-th component of v. Let $|v|$ be the ℓ_2 norm of v. For a matrix $M \in \mathbb{R}^{d \times d}$, let $\text{diag}(M)$ be the vector made with the diagonal elements of M. The k-th row of a matrix M is $\texttt{row}(M, k)$ and the column k is $\texttt{col}(M, k)$. The matrix $\mathbb{I}$ is the identity matrix of the pertinent size.

Let $z \in \mathbb{R}^d$ be a random variable, and $f : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ its probability density function (pdf). Let $\mathbb{E}(z) = \int z f(z) \mathrm{d}z$ be its expected value, and $\text{Cov}(z) = \mathbb{E}[(z - \mathbb{E}(z))(z - \mathbb{E}(z))^T]$ be its covariance. Given z with pdf $f(z)$ and a subset $\Omega \subseteq \mathbb{R}^d$, we define a conditional pdf $f_\Omega : \mathbb{R}^d \to \mathbb{R}$ as $f_\Omega(z) = f(z \,|\, z \in \Omega)$; that is, $f_\Omega(z) = \frac{f(z) 1_\Omega(z)}{P(\Omega)}$, where 1_Ω is the indicator function and P is the probability mass of Ω, which is computed using $f(Z)$. The conditional random variable with pdf f_Ω is referred to as the f-distributed random variable z with truncated support Ω. The one-dimensional normal probability density function, $\phi : \mathbb{R} \to \mathbb{R}_{\geq 0}$, is defined as $\phi(z) = \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2} z^2)$, $z \in \mathbb{R}$. The normal distribution's tail probability, that is, the probability that the random variable is larger than a quantity Z, is denoted by $\mathbf{Q} : \mathbb{R} \to [0, 1]$, and defined as $\mathbf{Q}(Z) = \int_Z^{+\infty} \phi(x) \mathrm{d}x$.

The state of the network at time step k, which is the same as the state of each one of the agents in the network in one big vector, is denoted by $\mathbf{x}^{all}(k)$. Vehicles in the network are identified by natural numbers, and their states are sorted according to their number in the network state vector. In a network made up of N agents, each agent will have a number i belonging to the set $\mathcal{G} = \{1, ..., N\}$. Agent i's state (composed by position and orientation) at time step k is denoted by $\mathbf{x}^i(k) = [x^i \quad y^i \quad \theta^i]^T$. Since this is not an estimate, but the true state, there is no covariance associated to it. The network state can be expressed as the vectors of all states in the network piled up:

www.manaraa.com

$$\mathbf{x}^{all}(k) = \begin{bmatrix} \mathbf{x}^1(k) \\ \vdots \\ \mathbf{x}^N(k) \end{bmatrix}$$

We denote agent i's estimate of the state of the network at time step k by

$$\hat{\mathbf{x}}_i^{all}(k) = [x_i^1 \quad y_i^1 \quad \theta_i^1 \quad ... \quad x_i^N \quad y_i^N \quad \theta_i^N]^T$$

and the associated covariance by $\mathbf{P}_i^{all}(k)$. For every other agent j that i communicates with, i keeps a mutual estimate denoted by $\hat{\mathbf{x}}_{ij}(k)$ and associated covariance $\mathbf{P}_{ij}(k)$. Agent i's estimate of agent's j state (that is, agent i's estimate of agent j's pose, composed by position and orientation) is denoted by $\hat{\mathbf{x}}_i^j(k)$, and the associated covariance by $\mathbf{P}_i^j(k)$. As a rule of thumb, one should think of the subscripts as denoting the owner of the estimate (or the vehicle where the estimates live in) and the superscripts as denoting the agent/s that are being estimated.

3.2 Problem statement

We now formalize the problem. Let us consider a team of N robots moving in a 2D non-changing environment $\mathcal{E}$. [1] Their dynamics (or motion model) are described by a discrete-time, time-varying system, such that at each time step k

$$\mathbf{x}^i(k+1) = f_i(\mathbf{x}^i(k), \mathbf{u}^i(k)) + \mathbf{w}^i(k) \tag{3.1}$$

where $\mathbf{x}^i(k) \in \mathbb{R}^d$ represents the state of robot i at time k, and $\mathbf{u}^i(k) \in \mathbb{R}^n$ is its control input. For all robots in the network, that is for $i = 1, ..., N$, the state $\mathbf{x}^i(k)$ is the main quantity that is being estimated, and is composed by the position and orientation of robot i. The process noise, $\mathbf{w}^i(k)$, is assumed to follow a normal distribution with zero mean and covariance $Q_i(k)$, and uncorrelated in time. For each robot pair in the network, their process noises are assumed to be uncorrelated with

[1] In our case, it does not matter if the environment is static or dynamic, since robots do not measure it in any way, nor do they build a map. In other scenarios the difference would be important, though.

www.manaraa.com

one another. Since we are working within a Kalman filter framework, Equation 3.1 is linearized to obtain

$$\mathbf{x}^i(k+1) = A_i(k)\mathbf{x}^i(k) + B_i(k)\mathbf{u}^i(k) + \mathbf{w}^i(k) \tag{3.2}$$

where $A_i(k) = \frac{\partial f_i}{\partial \mathbf{x}^i}(\hat{\mathbf{x}}_i^i(k)) \in \mathbb{R}^{d\times d}$ and $B_i(k) = \frac{\partial f_i}{\partial u_i}(\hat{\mathbf{x}}_i^i(k)) \in \mathbb{R}^{n\times d}$ are the Jacobian matrices. Here and onward, the hat symbol ($\hat{}$) over a variable represents an estimate after having been updated with data at the current time step, the bar symbol ($\bar{}$) represents an estimate before having been updated at the current time step, and no symbol represents the true value.

At time k, robot i can take a combination of m_i local and global measurements, which are stacked into $\mathbf{y}_i(k) \in \mathbb{R}^{m_i}$. Local measurements include range and bearing to other vehicles, whereas global measurements are absolute pose measurements from GPS. Let $\mathbf{x}^{all}(k) \in \mathbb{R}^{Nd}$ be the vector of all vehicles' states. The measurement model is

$$\mathbf{y}_i(k) = h_i(\mathbf{x}^{all}(k)) + \mathbf{v}_i(k) \tag{3.3}$$

This can be linearized, which yields

$$\mathbf{y}_i(k) = C_i(k)\mathbf{x}^{all}(k) + \mathbf{v}_i(k) \tag{3.4}$$

where $C_i(k) = \frac{\partial h_i}{\partial \mathbf{x}^{all}}(\hat{\mathbf{x}}_i^{all}(k)) \in \mathbb{R}^{m_i\times Nd}$. The measurement noise, $\mathbf{v}_i(k)$, is assumed to be normally distributed, with zero mean and diagonal covariance $R_i(k)$, and uncorrelated in time. The measurement noise associated with a robot is uncorrelated to the process noise, as well as to other robots' measurement noises.

In decentralized CL such as the one we propose here, agents share some or all their local measurements with neighboring agents in order to keep the trace of their covariance matrix as low as possible. [2] In a centralized approach, on the other hand, the measurements are not sent

[2] We are careful here not to talk about minimization, since this is not a formal minimization problem with a defined cost function where the primary goal is to have a covariance matrix trace that is as small as possible.

www.manaraa.com

between agents but to a fusion center for processing instead. An explanation of the main differences between centralized and decentralized approaches can be found in Section 3.4.

3.3 Why cooperative localization?

The problem has been now formally set up. In this section, we discuss more in detail what exactly makes CL an interesting approach that has been used extensively in the literature to solve the kinds of problems we are interested in, and what are possible downsides to using it.

The main strength of cooperative localization resides in the fact that agents can take relative measurements to one another and send them to other agents, thus making it possible to exploit the coupling that exists between agents states through the measurement model. In particular, sending relative measurements between agents causes the covariance matrix to become populated in its non-diagonal elements, so when one agent receives an absolute position measurement, the cross-correlations allow the network to improve their own state estimates as well. Let us now see a simple CL example.

Imagine we have a team of $N = 4$ robots on a plane that are not moving, as illustrated in Figure 3.1. Vehicle motion makes the problem harder to solve, but for the intents of this example it is not needed since it does not change the problem fundamentally. The pose of one arbitrary agent is assumed known, with some degree of uncertainty (we can imagine that this particular agent has a GPS receiver, or detects a feature in the environment that allows it to situate itself with respect to the global frame). However, for the rest of the agents, we have to rely on the measurements they take to find their poses.

Let us first assume a non-cooperative approach is being used. In this scenario, agents detect features or landmarks in the environment whose location is known in the global reference frame to localize themselves. In this case, it is easy to see that even if agents share these kinds of measurements with one another, there is zero benefit from doing so. In other words, an agent can not really use the information of where another agent is *to localize itself better*. This is a key aspect; information of where other agents are in the world may indeed be beneficial in a lot of applications.

www.manaraa.com

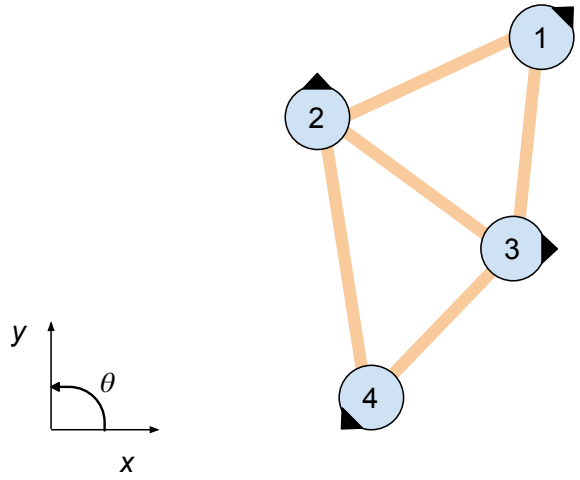


Figure 3.1: Example of CL. The 4 agents are static and have a specific position and orientation in the reference frame shown. Communicating agents are indicated by the yellow lines between them.

www.manaraa.com

However, it is not in the process of localizing oneself, since agents have no way of sensing each other. Hence, regardless of whether a centralized or decentralized approach is used, the estimates are computed with measurements from the agents to the environment.

On the other hand we have the cooperative way. Here, the vehicles take relative measurements to other vehicles. Vehicles can also, as seen in the non-cooperative approach, detect the environment, although in this case it is not necessary for all of them to do so (this is, in fact, one of the strengths of CL, as explained next). If we focus on what happens to an arbitrary agent, we see that these relative measurements that it is taking, by themselves, are not more useful than detecting features in the environment, as done in the previous case, when it comes to localizing itself. However, when all these measurements taken by all agents are put together, whether that is in a central estimator or by sharing them between agents, the problem will be highly constrained. In other words, the value of some observations will provide information about the value of other observations. Additionally, being able to pin down one of the agents will positively impact the whole network, since the relative configuration will be estimated with high accuracy.

The ability to *jointly* estimate the state of the network makes CL a method that is not only more accurate than other methods thanks to the coupling that is created between agents, but also more robust. In the event that an agent were to lose power, other agents would still be able to measure it and figure out where it is. Additionally, in CL sporadic access to accurate information by one of the team members results in a net benefit for the rest of the team.

There are other advantages associated with CL that are not directly related to estimation performance. One feature is that in general there will not be any data association problems, since the robots can be equipped with distinctive features or identifiers that other robots can use to uniquely identify them. Another positive aspect is the lack of need for existing infrastructure; cooperative localization can work using the usual operational components that robotic platforms are equipped with. Finally, CL can easily be used together with other types of measurements to increase the accuracy in the estimates, as we have briefly mentioned already. Sometimes, the lack of environmental features, low light conditions, low-quality and noisy odometric sensors and

www.manaraa.com

other elements will make estimation more challenging. In such cases, because CL does not depend on any of these, it can be use to augment the estimation capabilities of the system and improve performance.

There are downsides associated with the use of CL that we can not neglect. The main one is that there are higher communication, memory and processing costs that have to be dealt with. Given the limitations of hardware platforms at the moment, many approaches, including the work in this thesis, focus on thinking of new ways to implement a CL algorithm cost-effectively such that it can be used in affordable platforms with low power.

3.4 Centralized versus decentralized CL

There are two main approaches to the implementation of a cooperative localization algorithm: centralized (CCL) and decentralized (DCL).

In a **centralized** scheme, agents take relative measurements to each other and afterwards send them to what is called the fusion center (FC), which can be either a leader robot or a computer that is not physically part of the robot network. This fusion center gathers and processes information from all team members at every time step. Since the measurements are properly identified, the FC does not have to worry about double-counting or incorrectly fusing the data. Once the data at the current time step has been processed, the FC broadcasts the estimates back to the team. The centralized approach is generally more costly than the decentralized one; the processing costs are high, since *all* agents' measurements are taken into account in the fusion step. In most scenarios that is a problem, since all vehicles rely on their batteries. Communication costs will often be comparable to those for the decentralized approach; this is due to the fact that even though agents have to send their measurements only once every time step to the FC, this may be farther away than the average neighbor agent in the decentralized case. Additionally, the centralized scheme will be less robust than the decentralized counterpart, since a failure at the FC will compromise the entire network.

In a **decentralized** scheme, robots communicate and share their relative measurements. The

www.manaraa.com

data fusion process is, therefore, performed by each robot independently, based on all the data that it has gathered. In consequence, depending on the communication graph, the estimates computed by an arbitrary robot will be more or less accurate, since an agent may be communicating directly with another agent that has access to accurate localization information or, on the other hand, it may be several communication links apart. Positive aspects of DCL when compared to CCL are its lower processing costs, since vehicles will generally fuse only a subset of all the data gather by all agents; increased robustness, since each agent keeps its own estimate; increased flexibility, since an agent may be added to or taken out of the network at any moment. The big challenge in DCL is to design algorithms that maintain consistent estimates by keeping account of all correlations between the agents' states [24]. There are many different ways to do this, and we talk about some of them in the Chapter 2.

www.manaraa.com

Chapter 4

Proposed solution

In Chapter 3 we have formally stated the problem that we are interested in solving and seen a landscape of the main general ways to frame a solution. In this chapter, we address the particular solution that we have developed. We begin by describing the notation, which admittedly can get cumbersome at times, so that the reader can refer to it in later sections; we then outline and justify the assumptions we have taken while developing our method; finally, we give a detailed look at the algorithm itself, including boxes with pseudocode, diagrams and written explanations.

4.1 Assumptions

Cooperative localization is a vast and complex topic, and we think it is important to first establish the assumptions to properly set up our problem of interest. These assumptions are dictated, firstly, by our particular goals and what we are trying to accomplish, and secondly, by the current knowledge of the field. We strive to achieve a middle ground between being too optimistic and having a very hard problem to solve and being too conformist and not bringing any relevant contributions.

In decentralized CL, each robot has the ability to communicate with other robots in the network. A protocol is assumed to be running on the network that allows each robot to know which other robots it can communicate with. The subset of agents that i can communicate with is denoted as $\mathcal{N}_i$, and does not change in time. We assume that communications are bidirectional, that is, if robot i can communicate with robot j, then j can communicate with i.

www.manaraa.com

We first assume that each robot $i \in \{1, ..., N\}$ keeps an estimate of the states of all agents in the network (we also refer to this as the state of the network for brevity) and the associated covariance matrix at each time k, $\{\hat{\mathbf{x}}_i^{all}(k), \mathbf{P}_i^{all}(k)\} \in \mathbb{R}^{Nd}, \mathbb{R}^{Nd \times Nd}$. Additionally, for each other agent j that i shares measurements with, i keeps a common or mutual estimate and the associated covariance, $\{\hat{\mathbf{x}}_{ij}(k), \mathbf{P}_{ij}(k)\} \in \mathbb{R}^{Nd}, \mathbb{R}^{Nd \times Nd}$. These can be interpreted as agent i's estimate of agent j's estimate of the state of the network at time k, and their purpose is to be used as conservative estimates in the process of deciding what measurements to send from i to j, as will be detailed later. The reason these are conservative is that only measurements shared between i and j will be used to update them, whereas all measurements taken by i and received from other agents, explicitly and implicitly, will be used to update $\{\hat{\mathbf{x}}_i^{all}(k), \mathbf{P}_i^{all}(k)\}$ and $\{\hat{\mathbf{x}}_j^{all}(k), \mathbf{P}_j^{all}(k)\}$. Also, since all information exchanged between i and j (sent *and* received from both agents) is used to update these shared estimates, then $\hat{\mathbf{x}}_{ij}(k) = \hat{\mathbf{x}}_{ji}(k)$ and $\mathbf{P}_{ij}(k) = \mathbf{P}_{ji}(k)$ for all pairs i, j and all time steps.

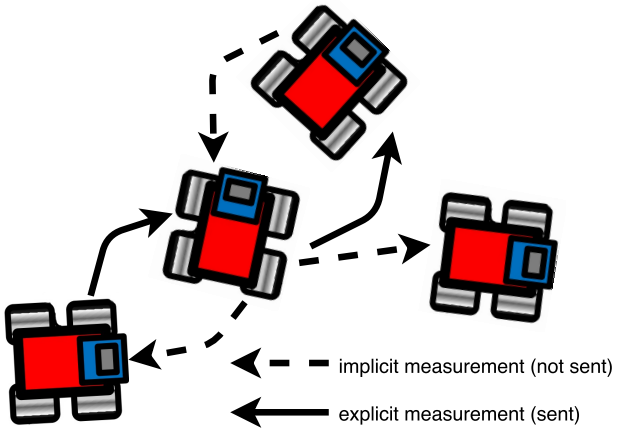


Figure 4.1: Representation of typical scenario where the robots exchange a combination of explicit and implicit information.

Agents move according to some laws and following the controls that we feed the system with. Since every agent keeps an estimate of the entire network's state, we assume that these control inputs for all agents are known to every agent. Agents also rely on relative measurements to other vehicles to compute their estimates. If vehicles' onboard clocks can be precisely synchronized, and

www.manaraa.com

their oscillators can maintain the time through the duration of the mission, then the relative range can be calculated using the one way travel time of flight [25]. Finally, it is assumed that the data transmitted between vehicles is always uniquely identified, and its origin is always perfectly known. Additionally, data can be dropped for different reasons, but a data packet that is destined to a particular agent will never be delivered to a different agent.

Assumptions summary

We now summarize the assumptions made for the reader's convenience:

- Communications between robots are bidirectional and instantaneous (zero latency).
- Agents' internal clocks are synchronized.
- Correspondence communications-measurements: if a robot can sense another robot, it can also share measurements with it.
- No data association problems: each data packet is uniquely identified and can not be mistaken for another packet.
- Non-changing neighbor sets: agents always communicate with the same subset of the team after deployment.
- Equivalent sensing, communication and processing capabilities among robots.
- Control inputs are known to all agents.

4.2 Proposed event-based algorithm

The algorithm proposed takes its main inspiration from [26], which in turn is an adaptation to a decentralized, multi-agent scenario of [22]. Our approach goes beyond the limitations of these works, which assume linear dynamics and measurements, being interesting on a conceptual level but lacking the applicability we are looking for. The key idea behind the method we propose here is the ability to trade-off between the quality of the estimates and the communication costs associated with sending measurements, by means of the innovation threshold parameter. At every time step, each agent i updates its own estimate of the current state of the network as well as its estimate of

www.manaraa.com

each other agent j's estimate of the network (where $j \neq i$ and $j \in \mathcal{N}_i$).

For a robot to minimize the communication cost of transmitting its measurements at every time step, it is desirable to employ an event-triggered strategy to be used within a Kalman filtering framework. As we have seen, agents have the ability to share measurements, although there is a cost associated with that which depends on the number of measurements sent. In order to reduce the communication costs associated with the sharing process, we propose an event-based strategy that allows us to send just the measurements (or more specifically, the components of the measurement vector) that are considered innovative enough for the receiving agent. Thus, at time k robot i will send the ℓ-th component of the measurement vector $\mathbf{y}_i(k)$, $\texttt{elem}(\mathbf{y}_i(k), \ell)$, only if the absolute value of the innovation for that component is larger than a fixed value, which we call the innovation threshold, δ. In other words, we know that measurements that have small innovations associated to them will not affect the state estimate by much. Expressed more formally, we will check if $\texttt{elem}(|\mathbf{y}_i(k) - h_i(\hat{\mathbf{x}}_{ij}(k))|, \ell)$ is greater than δ, and send $\texttt{elem}(\mathbf{y}_i(k), \ell)$ if the condition is met. If the condition is *not* met, then that component is not sent, and robot j receives an empty measurement. However, since the innovation threshold is known to all agents in the network, the absence of a measurement component gives implicit information about that very measurement component, and robot j is able to fuse that information into its state estimate using a modified Kalman filter update.

The following sections provide detailed information about our proposed method. Figure 4.2 shows a block diagram that is meant to help in acquiring a conceptual understanding of the algorithm.

4.2.1 Observations

In our model, the agents that any given agent can measure (and communicate with) are fixed and do not change during the execution of the algorithm. Another possible model is that in which the agents can only communicate with neighbors that are within a certain radius from the measuring agent. Although both are perfectly valid, we find the former more useful when studying

www.manaraa.com

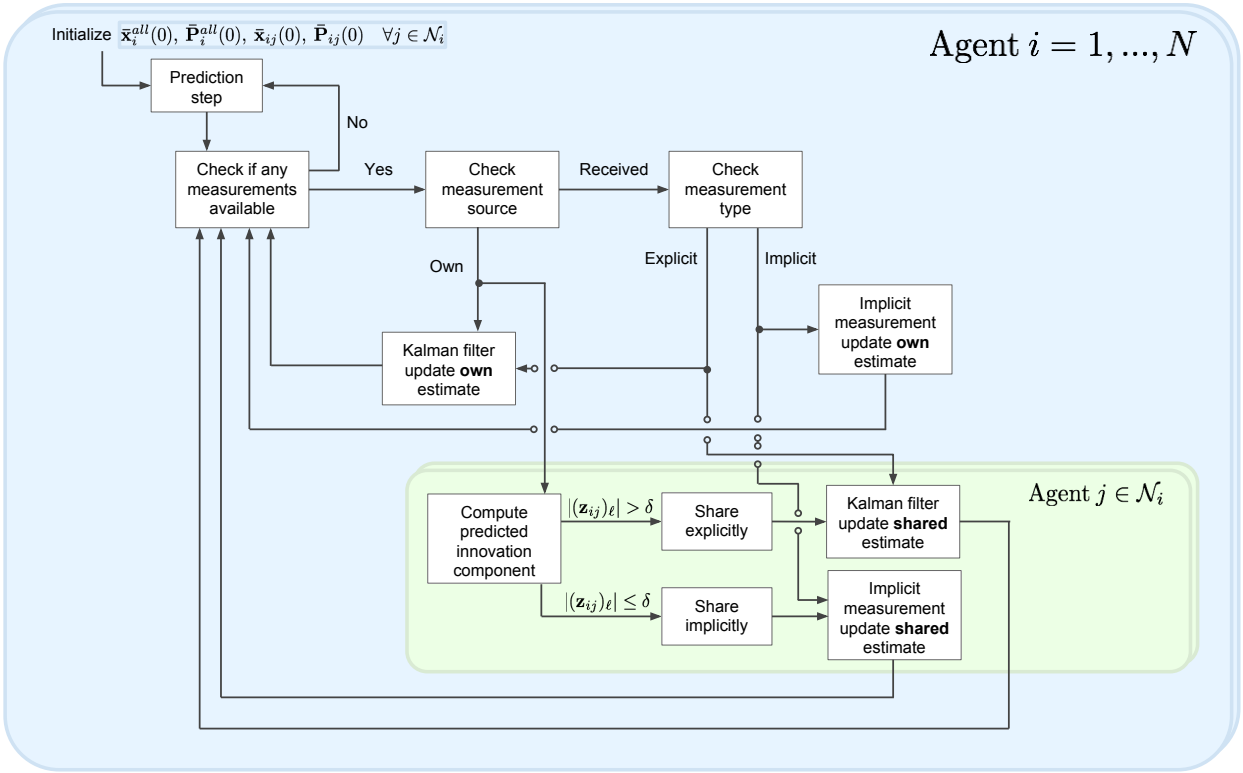


Figure 4.2: Block diagram showing the fundamental elements of the event-based algorithm

www.manaraa.com

the performance of our event-based algorithm, since we can have control over the communication graph regardless of the motions of the agents and the trajectories they follow.

The agents, hence, move describing certain paths in an environment. The environment, even though can have objects, walls, borders and additional features, is not observed by the agents, that is, agents can just observe other agents. We assume there are no beacons or landmarks for the agents to position themselves, and agents have to rely on relative measurements and rare GPS measurements. At each time step k, each robot i obtains a series of observations from its sensor suite that are grouped to form the measurement vector $\mathbf{y}_i(k) \in \mathbb{R}^{m_i}$. This vector will be used by i to update its state estimate. Also, after some processing, a modification of vector will also be shared with all agents $j \in \mathcal{N}_i$.

4.2.2 Communications

Once each agent i has gathered the measurement vector at the current time step it will run a series of processes to determine what particular components should be sent to the rest of the agents that i communicates with. It will do that by using its *a priori* common estimate with each other agent $j \in \mathcal{N}_i$ to determine which, if any, components of its measurement vector to send to each neighbor based on the innovation threshold parameter δ. [1] The importance of these shared common estimates is crucial; the process to decide whether to send a specific component of the observations vector could be performed using each agent's own estimate. However, the shared estimates are specifically tailored for that purpose, since they are updated only with the measurements exchanged between the two agents that share them, hence being more conservative and leading to the triggering condition being met in the right circumstances. After the components that will be shared explicitly and implicitly have been determined, the measurements are sent directly to the robot that is meant to receive them. The measurement vector sent by agent i to agent j will be different, in general, for each j, and different as well from the original measurement

[1] Nothing forces the innovation parameter to be the same across the network, although that is the case in all simulations and analyses in this work, hence the use of δ instead of δ_i.

www.manaraa.com

vector, $\mathbf{y}_i(k)$. As a side note, we do not consider the possibility that the messages can be sent to the wrong agent, nor are there data association problems that can lead agents to think that a data packet comes from an agent that is not the original sending agent.

The complementary process to sending measurements is receiving them. Since communications are assumed to be symmetric, for each agent j that i sends data to, it will also receive data from. Agent i will process two kinds of measurements. First, those from its sensor suite, which will be used to perform a Kalman filter update to its own posterior state estimate. Second, those it receives from other agents in either explicit or implicit form. Implicit measurements are defined to be the knowledge gained when a measurement is not sent (i.e. when the innovation was smaller than the threshold). Explicit measurements are fused using a Gaussian measurement model via a Kalman filter update, while implicit measurement updates are performed using Gaussian moment matching approximations for set-based measurement updates. *All* measurements are used to update $\hat{\mathbf{x}}_i^{all}(k), \mathbf{P}_i^{all}(k)$, whereas only measurements sent between i and j (explicitly and implicitly) are used to update $\mathbf{x}_{ij}(k), \mathbf{P}_{ij}(k)$. While the shared estimates are not what we are looking for per se, they are necessary to determine which measurements will be shared between any two communicating agents.

4.2.3 Measurement update

Agent i, after sending the measurements it has taken with its sensor suite to its neighbors (agents that it communicates with), will check its inbox to get the measurements its neighbors have sent to it and process them. As we have seen, these will be a combination of explicit and implicit measurements. Explicit measurements will show as the actual value of that measurement component, whereas implicit ones will show as a non-existing value (which is different than having a measurement that takes the 0 value). Explicit and implicit data will be fused differently, as the next two sections detail.

Explicit information fusion

www.manaraa.com

Each explicit measurement component ℓ agent i receives from each other agent $j \in \mathcal{N}_i$, and the row of the observation model Jacobian that corresponds to the ℓ-th measurement component, $\texttt{row}(C_i(k), \ell)$, will be used to perform an element-wise Kalman update on its own estimate, $\{\hat{\mathbf{x}}_i^{all}(k), \mathbf{P}_i^{all}(k)\}$, as well as on the corresponding estimate between i and j, $\{\mathbf{x}_{ij}(k), \mathbf{P}_{ij}(k)\}$, as Algorithm 1 shows. Additionally, from the measurement vector taken by i (that is, not received from other agents), the whole vector will be used to update its own estimate and covariance, whereas only the components shared explicitly between i and each other agent $j \in \mathcal{N}_i$ will be fused explicitly into the shared estimates.

Algorithm 1 Extended Kalman Filter Measurement Update

Require: $\hat{\mathbf{x}}(k), \mathbf{P}(k), C(k), R, \mathbf{y}(k)$
1: $K = \mathbf{P}(k)C(k)^T(C(k)\mathbf{P}(k)C(k)^T + R)^{-1}$
2: $\mathbf{z} = \mathbf{y}(k) - h(\hat{\mathbf{x}}(k))$
3: $\hat{\mathbf{x}}(k) = \hat{\mathbf{x}}(k) + K\mathbf{z}$
4: $P(k) = (\mathbb{I} - KC(k))\mathbf{P}(k)$
5: **return** $\hat{\mathbf{x}}(k), \mathbf{P}(k)$

Implicit information fusion

As mentioned before, an implicit measurement looks to the receiving agents as if nothing was sent for a specific component (which is known) of the measurement vector. Similarly to the explicit fusion process, each implicit measurement component ℓ agent i receives from each other agent $j \in \mathcal{N}_i$ is used to perform an implicit update on its own estimate, $\{\hat{\mathbf{x}}_i^{all}(k), \mathbf{P}_i^{all}(k)\}$, as well as on the estimate that is shared between i and j, $\{\mathbf{x}_{ij}(k), \mathbf{P}_{ij}(k)\}$. The components shared implicitly between i and each other agent $j \in \mathcal{N}_i$ will also be fused implicitly into the shared estimates. This brings about savings in communication costs, since there is no need for any bits to be allocated on an implicit measurement. Information extraction from this lack of explicit data is still possible thanks to the knowledge by all agents of the innovation parameter.

The derivation of the mathematical expressions that let us fuse implicit information together with explicit information into our estimates consists of two main problems that we cover next.

Bayes' theorem allows us to express the posterior distribution as a function of the prior

www.manaraa.com

distribution and a likelihood model. In the case of a general state estimation problem, we can write

$$p(x(k)|y(k),...,y(1),u(k),...,u(1)) =$$

$$= \frac{p(x(k)|y(k-1),...,y(1),u(k),...,u(1))p(y(k)|x(k),y(k-1),...,y(1),u(k),...,u(1)}{p(y(k)|y(k-1),...,y(1),u(k),...,u(1))} \tag{4.1}$$

And the posterior distribution will be Gaussian if both the prior and the likelihood function are Gaussian too.

In the case of implicit measurement fusion, we only know the measurement is within some boundaries, i.e. $y \in \Theta$, not its exact value. In other words, it is not point-valued but set-valued. Therefore, we write Bayes' theorem as

$$p(x(k)|y(k),...,y(1),u(k),...,u(1)) =$$

$$= \frac{p(x(k)|y(k-1),...,y(1),u(k),...,u(1)) \int_\Theta p(y(k)|x(k),y(k-1),...,y(1),u(k),...,u(1)\mathrm{d}y}{\int_\Theta p(y(k)|y(k-1),...,y(1),u(k),...,u(1))\mathrm{d}y} \tag{4.2}$$

However, here the posterior will *not* be Gaussian. The proposed approach operates under the assumption that this distribution can be approximated by a Gaussian with its same mean and covariance.

The previous assumption allows us to formulate the problem recursively and within a Kalman filter framework (see Appendix in [22] for formal proofs). However, when processing implicit measurements we need to deal with truncated Gaussian distributions for which we need to calculate mean and covariance. This is due to the fact that the likelihood function in Equation 4.2 is an integral over the set of possible values y can take, Θ. Fortunately, the first and second moment evaluations of truncated Gaussian distributions have analytic expressions and have been studied extensively, for example in [27, 28]. These results enable the fusion of measurements that are both explicit and implicit into our estimates. Algorithm 2 shows the computational steps needed to

www.manaraa.com

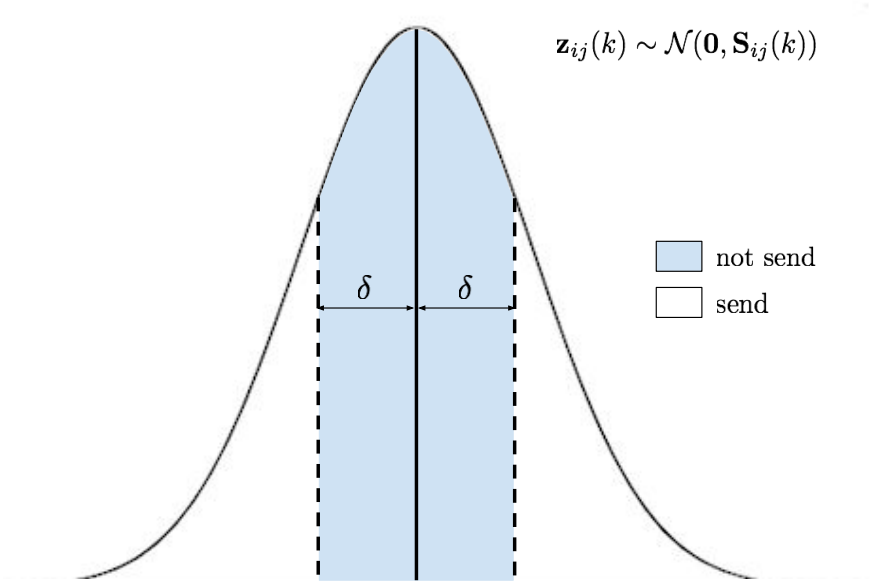


Figure 4.3: If the innovation associated to a measurement falls in the blue region, that measurement will not be sent, but rather shared implicitly. Innovations that fall in the white region are considered large enough to be sent.

perform implicit measurement fusion. This shows the estimates for a generic agent, and $\mathbf{x}_{\text{ref}}$ is the common or shared estimate with the corresponding communicating agent.

Algorithm 2 Implicit Measurement Update

Require: $\bar{\mathbf{x}}(k), \bar{\mathbf{P}}(k), \hat{\mathbf{x}}(k), \mathbf{P}(k), \mathbf{x}_{\text{ref}}, C(k), R, \delta$

1: $\mu = C(k)(\hat{\mathbf{x}}(k) - \bar{\mathbf{x}}(k))$
2: $\alpha = C(k)(\mathbf{x}_{\text{ref}} - \bar{\mathbf{x}}(k))$
3: $Q_e = C(k)\bar{\mathbf{P}}(k)C(k)^T + R$
4: $\bar{\mathbf{z}} = \frac{\phi(\frac{-\delta+\alpha-\mu}{\sqrt{Q_e}}) - \phi(\frac{\delta+\alpha-\mu}{\sqrt{Q_e}})}{\mathbf{Q}(\frac{-\delta+\alpha-\mu}{\sqrt{Q_e}}) - \mathbf{Q}(\frac{\delta+\alpha-\mu}{\sqrt{Q_e}})}\sqrt{Q_e}$
5: $\vartheta = \left(\frac{\phi(\frac{-\delta+\alpha-\mu}{\sqrt{Q_e}}) - \phi(\frac{\delta+\alpha-\mu}{\sqrt{Q_e}})}{\mathbf{Q}(\frac{-\delta+\alpha-\mu}{\sqrt{Q_e}}) - \mathbf{Q}(\frac{\delta+\alpha-\mu}{\sqrt{Q_e}})}\right)^2 - \frac{(\frac{-\delta+\alpha-\mu}{\sqrt{Q_e}})\phi(\frac{-\delta+\alpha-\mu}{\sqrt{Q_e}}) - (\frac{\delta+\alpha-\mu}{\sqrt{Q_e}})\phi(\frac{\delta+\alpha-\mu}{\sqrt{Q_e}})}{\mathbf{Q}(\frac{-\delta+\alpha-\mu}{\sqrt{Q_e}}) - \mathbf{Q}(\frac{\delta+\alpha-\mu}{\sqrt{Q_e}})}$
6: $K = \mathbf{P}(k)C(k)^T(C(k)\mathbf{P}(k)C(k)^T + R)^{-1}$
7: $\hat{\mathbf{x}}(k) = \hat{\mathbf{x}}(k) + K\bar{\mathbf{z}}$
8: $\mathbf{P}(k) = (\mathbb{I} - \vartheta K C(k))\mathbf{P}(k)$
9: **return** $\hat{\mathbf{x}}(k), \mathbf{P}(k)$

www.manaraa.com

4.2.4 Covariance intersection

In specific scenarios, it may be desirable to keep the uncertainty in the estimates bounded so that we get an idea of how large the errors are, provided that the filter is consistent. For certain communication graphs it may happen that, even if the covariances start off at small values, the lack of information about an agent to another agent causes the associated elements in both covariance matrices to grow indefinitely. That will happen, in fact, whenever any two agents can not measure each other and there is no third agent that can act as a link by measuring both of them and sharing those measurements with them. To deal with this problem, we introduce an event-triggered covariance intersection fusion method as a way for agents to improve their estimates of other agents that they cannot receive measurements about.

The covariance intersection algorithm for data fusion was first introduced in [17]. In contrast to the measurement updates we perform using the Kalman and implicit methods we have already presented, covariance intersection (CI) offers a way to fuse two state estimates into a new state estimate which bears the information contained in the two original estimates and their associated covariance matrices (see Figure 4.4).

Covariance intersection requires a parameter $\omega \in \mathbb{R}_{\geq 0}$ that determines the relative weight that each of the two estimates has in the computation of the final estimate, based on the quality of each estimate. Then, this parameter is solved for in an optimization process in which the cost function is the trace of the resulting covariance matrix, which of course, depends on ω. Algorithm 3 describes this process in more detail.

Some of the characteristics of CI are its ability to preserve **consistency** in the filter and the facts that it is a **conservative** method, **prevents double-counting** of measurements, and it is done in an ad-hoc manner between any two agents. However, despite being a convenient tool in certain situations, there are certain issues associated with covariance intersection that make it impractical as the primary estimate update method. An example of an algorithm that uses CI as its main update method can be found in [16], where agents take relative measurements to neighbors

www.manaraa.com

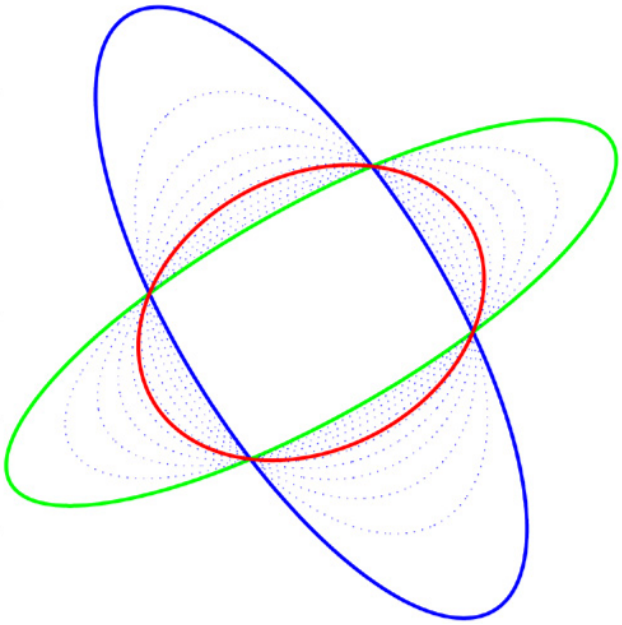


Figure 4.4: Example of covariance intersection in 2D. The blue and green ellipsoids represent the two original covariances, whereas the red ellipsoid represents the final covariance after the fusion, corresponding to the ω parameter that optimizes the problem. The dotted lines are solution candidates. Modified from [1].

and then update their own estimates without sending or receiving other agents' measurements. The cross-correlations between agents' states are introduced by the CI method through the combination of the respective covariance matrices. However, there are many drawbacks when doing this. One is that sending estimates (which include the estimated state and associated covariance) over the communication channel is more energy-consuming than it is to send measurements to other agents [26]. Second, the costs of computation are greater than when a Kalman filter update is performed, due to the additional optimization problem that needs to be solved in order to find the weighting parameter (see Algorithm 3). Finally, in decentralized scenarios CI produces suboptimal fusion results [18], since it is a conservative process that discards new information. On the other hand, the method we propose here can achieve the performance of a centralized algorithm more closely.

www.manaraa.com

Event-triggered CI algorithm

Algorithm 3 Covariance Intersection Algorithm

Require: $\mu_1, P_1, \mu_2, P_2, \beta$

$\omega^* = \text{argmin}_{\omega \in [0,1]} \text{trace}((\omega P_1^{-1} + (1-\omega)P_2^{-1})^{-1}\text{diag}(\beta))$

$P_{\text{new}} = (\omega^* P_1^{-1} + (1-\omega^*)P_2^{-1})^{-1}$

$\mu_{\text{new}} = P_{\text{new}}(\omega^* P_1^{-1}\mu_1 + (1-\omega^*)P_2^{-1}\mu_2)$

return $\mu_{\text{new}}, P_{\text{new}}$

4.2.5 Prediction

After each vehicle has fused the measurements from its own sensor and those received from neighbor agents and, if the conditions are met, after covariance intersection has been performed, all team members will proceed to propagate the state estimates and covariances forward in time, until the next measurement is taken and the estimates can be updated again.

The estimate for an arbitrary vehicle's state at time step k will be denoted by $\hat{\mathbf{x}}(k)$, and consists of a 2-dimensional position and orientation vector in a global reference frame for said vehicle.

$$\hat{\mathbf{x}}(k) = [x \quad y \quad \theta]^T \tag{4.3}$$

The predicted estimate for an arbitrary vehicle's state at time step $k+1$ will be denoted by $\bar{\mathbf{x}}(k+1)$:

$$\bar{\mathbf{x}}(k+1) = [x' \quad y' \quad \theta']^T \tag{4.4}$$

The notation here is purposely generic, since we are not modeling a particular agent but rather the state of any vehicle in the network.

For all team members, the Dubin's motion model (or velocity motion model) is used, which assumes we can control a vehicle through two velocities, a translational one, v and a rotational one,

www.manaraa.com

ω. Then, the control input vector is at time step k is

$$\mathbf{u}(k) = [v \quad \omega]^T \tag{4.5}$$

We establish that a positive rotational velocity induces a counterclockwise rotation from a top view of the plane, and a positive translational velocity induces a forward motion. We assume that the dynamics and control inputs are perfectly known within each agent, with the exception of some process noise that acts on all vehicles in the same way but is not modeled. Equation 4.6 expresses the predicted state of a vehicle at time step $k+1$ as a function of the state at time k assuming $\omega \neq 0$:

$$\begin{aligned} x' &= x - \tfrac{v}{\omega}\sin(\theta) + \tfrac{v}{\omega}\sin(\theta + \omega \mathrm{d}t) \\ y' &= y + \tfrac{v}{\omega}\cos(\theta) - \tfrac{v}{\omega}\cos(\theta + \omega \mathrm{d}t) \\ \theta' &= \theta + \omega \mathrm{d}t \end{aligned} \tag{4.6}$$

Vehicles can use its own processors to integrate the motions of all team members to compute the prior estimates and covariances for the next time step. Algorithm 4 details how this is done for *one* vehicle at a time. In other words, Algorithm 4 is run within each agent for *every* team member at each time step in which we are predicting the state of the network for the next time step. Therefore, the inputs $\hat{\mathbf{x}}(k)$ and $\mathbf{u}(k)$ are generally different for every vehicle's state we are running the algorithm for.

Algorithm 4 Time Update Algorithm

Require: $\hat{\mathbf{x}}(k) = [x \quad y \quad \theta]^T, P(k), Q(k), \mathbf{u}(k) = [v \quad \omega]^T$
1: Solve Equation 4.6 for $\bar{\mathbf{x}}(k+1)$
2: Solve Equation 4.8 for Φ and Γ
3: Solve Equation 4.7 using Φ and Γ for $\bar{\mathbf{P}}(k+1)$
4: **return** $\bar{\mathbf{x}}(k+1), \bar{\mathbf{P}}(k+1)$

On top of predicting the state estimates for all agents at the next time step, the associated covariance matrix needs to be determined as well. We know that it can be computed by means of

www.manaraa.com

the following expression:

$$\bar{\mathbf{P}}(t) = \Phi(t,k)\mathbf{P}(k)\Phi^T(t,k) + \Gamma(t,k)Q(k)\Gamma^T(t,k) \tag{4.7}$$

where $\Phi(t,k) \triangleq \frac{\partial \mathbf{x}(t)}{\partial \mathbf{x}(k)}$ is the state transition matrix between the initial time step k and the final desired time t for each agent's state, and $\Gamma(t,k) \triangleq \frac{\partial \mathbf{x}(t)}{\partial \mathbf{w}(k)}$ is the process noise transition matrix. These are computed solving the following system of differential equations:

$$\begin{aligned} \dot{\Phi}(t,k) &= A(t)\Phi(t,k) \\ \dot{\Gamma}(t,k) &= A(t)\Gamma(t,k) + D \end{aligned} \tag{4.8}$$

where $A(t) = \frac{\partial f}{\partial \mathbf{x}(t)}(\hat{\mathbf{x}}(k))$ is the Jacobian of the corresponding agent's dynamics, and $D = \mathbb{I}_{3\times 3}$. Analytic solutions to this system in general do not exist, so it is usually solved by numerical integration.

www.manaraa.com

Chapter 5

Simulations

We now present the organization of the simulations that have been performed to test the proposed event-based estimation algorithm.

To this day, event-based algorithms using the implicit measurement update that our algorithm uses have not considered non-linear effects, hence not using realistic measurement and dynamics and controls models. The work carried out for this thesis emerges as the natural next step to take after previous work done by other members in academia and industry with linear systems, given that one of the main goals of the community working on robot localization is to study the applicability of these algorithms to realistic scenarios. Whenever there is a new method whose performance we want to compare to other state-of-the-art methods and whose characteristics are poorly known, simulations can offer clear insight. Computer simulations are often a vital tool by which to predict, to an extent that will depend on our ability to model and represent a part of the real world (our system), the behavior of the actual platform we are trying to design and build. Hence, the main motivation behind simulations is to somehow be able to study all the different phenomena that are happening simultaneously and achieve a better understanding of what parameters trigger what behaviors in our algorithm. Even though they are not sufficient, in and of themselves, in this work we understand that the results obtained are only a first approximation towards successfully implementing our algorithm on a real system.

These simulations are by themselves one of the contributions of this work, since the results throw a lot of light not also to motivating the interest of the proposed method, but also to thinking

www.manaraa.com

about future research activities.

We need to be precise when talking about comparing the performances of different Kalman-based filters. Here, by performance we refer to a series of characteristics of the filter that make it more desirable to be implemented in *specific configurations* than other filters. A narrow look at the performance would only consider aspects such as state errors and consistency. However, we understand that the design of an estimation algorithm for multi-robot localization has to take into account other features, such as associated communication costs, computational complexity, robustness or scalability. Accordingly, a view that tries to be as holistic as possible (with the obvious limitations associated with limited time and scope) is adopted.

The performance of the algorithms will obviously be dependent on the values of the system parameters and the specific conditions of use. Hence, we need to do a study that considers the most relevant parameters and how the estimation capabilities change as these parameters take on different values.

Our goals, then, are two-fold:

(1) Identify all the relevant user- and environment-dependent parameters that, when altered, translate into changes in the performance of our filter, as measured by pre-established metrics and indicators. These can be knobs that are either chosen by the user at the time of setting up the fleet of robots for deployment, like the innovation threshold, imposed by the environment, like the communication probability, or imposed by the hardware platform, like sensor noise or processor speed.

(2) Perform an exhaustive analysis of the performance of our event-based filter with the purpose of using the results to provide guidelines of usage. More specifically, the goal is to determine a set of specific conditions under which this algorithm can be used with an expected accuracy and consistency. This will be explored in order to get a thorough insight into the intrinsic characteristics of the event-based filter and to derive more generalist trends which are situation-independent.

www.manaraa.com

5.1 Study parameters

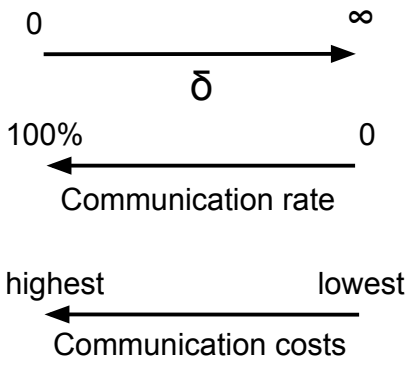
We begin by identifying the key parameters of our problem. Arguably, the most important and distinctive aspect of our event-based filter is the **innovation threshold**, δ, which is used as a tool to regulate the amount of bits in the data packets that vehicles exchange in their attempt to find the state of the network. The larger this threshold is, the more "innovative" the measurements will have to be in order to be shared explicitly with other agents and the lower communication costs will be, as Figure 5.1a shows. However, this comes at a cost – there is obviously a tradeoff between the amount of messages that are shared explicitly and how the filter performs, at least intuitively. When measurements roam within the two innovation bounds set by δ, the implicit update will have a smaller impact on both the state estimate and covariance than the explicit update would. The lack of a received explicit measurement, combined with the vehicles' knowledge of the value of δ, allows team members to extract *some* information from these situations, but the amount of knowledge will decrease as the innovation threshold becomes larger. In the limit case, where δ would be arbitrarily large, no measurements would be explicitly shared at all, and the filter would eventually diverge.

By scrutinizing the performance of the filter under a wide range of innovation threshold values, our intention is to gather the necessary information to determine what is happening and why it is happening.

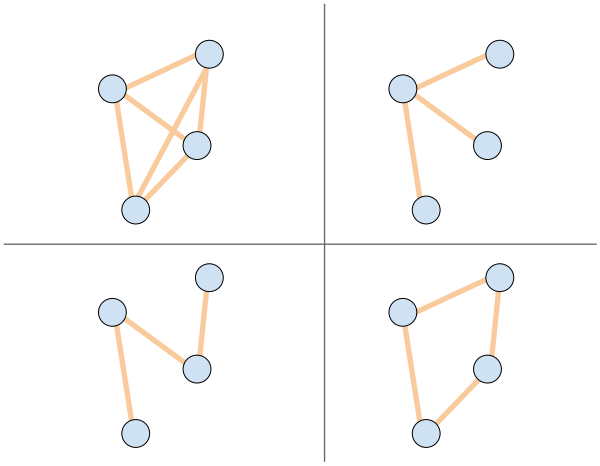
While intuition is an important notion in general, in science it needs to always be backed by evidence. Even though it seems natural to think that the larger the innovation threshold, the larger the mean value of the error, this has to be verified. Additionally, intuition offers a good starting point for study, but the specific questions we want to answer are far more elusive. Here, we are not only interested in sketching out the isolated effects of increasing or decreasing the innovation threshold, but also on finding numbers to perform a quantitative analysis and, especially, on how this relates to the bigger problem we are trying to tackle, where many other aspects come into play. It is for this reason that the analysis performed here considers the interaction between different

www.manaraa.com

parameters.



(a) Here we see the qualitative relationship between innovation threshold, communication rate and communication costs. The direction of the arrows represents an increase in the associated quantity.



(b) Possible communication topologies. The number of links between agents, and the access of these agents to privileged information, are elements that affect the performance achieved in the estimates.

Figure 5.1

When thinking about realistic scenarios (those in which multi-agent systems usually have to carry out their tasks), there are some changes we need to include in our models with respect to idealistic or more simplified cases. One issue encountered very often in real-life applications that can seriously compromise the performance of the proposed event-based filter is that of imperfect communications. There may be several causes for this, from walls or other objects interfering with the signal, to low reliability due to environment type and bandwidth limitations. Regardless of the nature of this phenomenon, we think it is crucial to account for these imperfections due to the potential impact they have on filter performance. There are two main sources of errors. Firstly, our algorithm, by definition, is a method for cooperative localization, which implies that it relies heavily on agent-to-agent measurements and communications. By compromising these, the filter loses to some extent its strength, which is derived from the correlations between vehicles' states. Secondly, our particular approach uses implicit information (that is, the assumption that a measurement that has not been received was *intentionally* not sent), which makes it vulnerable to lost measurements.

www.manaraa.com

Essentially, this means the receiving agent will incorrectly fuse a lost measurement as an implicit one, leading to errors and inconsistencies in the estimates. In all the different simulations, the reach of these effects is what we pursue to know. Ultimately, we want to answer questions such as how safe is it to use our event-based filter in low-reliability scenarios (e.g. underwater), how quickly and abruptly do estimates degrade, or if there are any preventive measures to take to shield against negative effects, for instance adaptive thresholding. For the reasons stated above, the second parameter studied will be the **communication success probability**.

To motivate the decision for the third study parameter, let us imagine two different scenarios. In the first one, a team of ground robots carries a heavy metal plate used in a construction project. The vehicles move slowly and, whenever possible, in a straight line, since dropping it could result in damage to the plate, robots, other elements or even people. In the second scenario, a team of UAV's is on a search and rescue mission in a forest. The motion here is faster and less predictable than in the first case, due to the nature of the mission and the capabilities of the hardware platform. In both scenarios, the robots cooperate with one another, and they could be using our event-based algorithm to incur in fewer communication costs. However, if we assume that the clocks of the robots in both cases operate at the same frequency, and that measurements and communications happen at that same frequency, it seems clear that the second scenario will be more challenging. In order to see the performance of our filter and how much it changes in different situations, the **type of motion** will be studied in the simulations.

Finally, we know cooperative localization happens, by definition, in teams of robots that measure and communicate with one another. With the exception of the 2-agent case, where the vehicles either communicate or not, in any multi-agent system there are different ways to establish the communications between agents. This is what we know as the **communication topology**, that is, the model that specifies which vehicle talks to which at any given time. While some topologies may be more adequate for cases where the goal is to compute estimates very accurately, there is a trade-off, again, between the amount of data that is transmitted and the performance achieved. In short, what different topologies will do is allow for information to spread faster in the network;

www.manaraa.com

hence, if a vehicle can pin down its own location, the vehicles that communicate with it directly will gain the benefits as well. In turn, other vehicles that communicate with these vehicles will be able to benefit, although less directly. In the same way that with the innovation threshold we attempt to be more energy-efficient by just using valuable information, here too some topologies may be better than others, and this is what the simulations will explore.

A few examples of communication topologies can be seen in Figure 5.1b. The graph in the top left is a case of "all-to-all", where each vehicle communicates with every other team member; although ideal for estimation performance, this approach is not very energy-efficient since a lot of power goes into transmitting the data to neighbors. Additionally, it also scales very poorly as the team size increases. Top right shows a graph where one of the agents acts as a hub, and the others rely on this central agent; this is a convenient configuration when one of the vehicles receives accurate information about its location, since it benefits all the team immediately, although it is quite sensitive to communication failures. Bottom left shows a line graph; in some practical situations where robots can only transmit data and detect up to a certain distance and the robots are physically semi-aligned, this may be the configuration we have to use. The main challenge about this graph is that information can take a long time to travel from edge to edge if the network is large. Finally, bottom right shows a circular diagram, which is similar to the line graph but with no edges; the behavior of the last two would in general be similar, although the travel distance would be significantly reduced in the latter, if we assume the same network size.

5.2 Performance metrics

When it comes to evaluating the performance of our algorithm, some metrics need to be chosen and defined. This section outlines the fundamental aspects that are used in the performance analysis of our algorithm, although other, more circumstantial, aspects that will be defined and explained as needed later may also be considered on top.

One of the most relevant quantities is the **mean squared error** (MSE), since it gives us an idea of how large the errors are on average. Here, we will use the MSE as a way to compare

www.manaraa.com

different algorithms and parameter combinations, focusing more on the relative differences than in the values themselves. [1]

$$\text{MSE} = \frac{1}{n}\sum_{s=1}^{n}(\hat{\mathbf{x}}_s - \mathbf{x}_s)^2 \tag{5.1}$$

where $\hat{\mathbf{x}}_s$ is the predicted at the time step of interest for simulation s, $\mathbf{x}_s$ is the true value at the same time step for the same simulation, and n is the total number of independent simulations.

The second metric we will consider is the **covariance matrix**, in particular the diagonal elements. This will not only provide an idea of the level of uncertainty in our estimates, but also, when compared to the MSE over a sufficiently large number of simulation runs, it will tell us how well the two match and, thus, how consistent the estimates are.

Another quantity used will be the **communication rate**, which relates the number of explicit measurements received with the total number of measurements. The communication rate (or CR) is a defined as the ratio between the total number of measurements that a given agent has sent to another agent and the total number of measurements that the sender has taken. [2] We note that the CR is not a rate in a time-wise sense, or a measure of *how often* data packets are sent. Rather, the communication rate is meant to be a direct measure of the amount of bytes sent and, in turn, of the power used by robots in doing so. Hence, the communication rate between agents i and j is:

$$CR_{ij} = \frac{\text{sent}}{\text{taken}} \tag{5.2}$$

This value is an indicator of *how much* information an agent is trying to communicate with another agent, and is a direct measure of communication costs.

Along the same lines, we will also make use of the transmission rate, which is different to the communication rate. The transmission rate is defined as the ratio between the total number

[1] Of course, keeping an eye on the values is always crucial, and we must make sure that they make sense in the context of our problem. In the simulations, we work with ground robots that move at speeds of 1 m/s and that are separated by distances of 15 m at most. With this information and other problem parameters, we can get an approximation of the errors we should be getting.

[2] We make a distinction between taking a measurement and receiving a measurement. The first one refers to all measurements that a vehicle gets from its sensor suite, and in general include GPS and range/bearing. The second one refers to all measurements that a vehicle has received from other agents through the communication channel.

www.manaraa.com

of measurements that an agent *receives* from another agent and the total number of measurements that the sender has taken. The transmission rate between agents i and j is:

$$TR_{ij} = \frac{\text{received}}{\text{taken}} \tag{5.3}$$

In perfect communications scenarios, these two quantities are always the same, since no measurements are dropped or lost. However, the distinction is crucial in our case due to the fact that we work with simulations in which data packets can be *sent* but not *received*, and the study of the algorithm performance has to take that into consideration. Another remark is that both communication and transmission rate are defined between two agents. This is different for the innovation threshold, which is defined for a particular agent regardless of other team members.

Finally, we note that the communication rate depends on the value of δ; although there is a clear relationship between the two (an increase in the innovation parameter will generally cause a decrease in the communication rate, which is the main metric we use to quantify communication costs), in our approach the former is computed heuristically based on the results which have the latter as one of the parameters. In other words, we fix δ and *a posteriori* we compute the CR using the simulations' results. The values of the CR as a function of δ will be problem-dependent, and accordingly, for a new problem, we will have to try a range of innovation threshold values and build the function heuristically. An analytic expression relating these two quantities for a much simpler system, with one sensor and one estimator and where all the processes are linear, can be found in [21].

5.3 Simulations setup

As described in the previous section, 4 main study parameters have been chosen: innovation threshold, communication success probability, type of motion, and communication topology. For each of these 4 parameters, a grid representing a discrete set of values will be created. The values in this grid are not selected randomly, but rather to cover the widest possible range of practically

www.manaraa.com

sound situations and applications. For example, when studying the effect of the motion, we will just consider dynamic models that are attainable for commercial robots, or when considering innovation threshold values we will not reach the point where no messages are sent at all. This ensures that as little computer time as possible is spent on non-relevant simulations.

For the innovation threshold, in the 2-agent simulations the values chosen are

$$\delta_{all} = \{0, 0.05, 0.11, 0.17, 0.25, 0.31, 0.40, 0.60, 0.85, 1.15, 1.50\}$$

δ adopts the units of the innovation, which implies it has no specific units (for the range component it will have distance units, for the bearing component it will have angle units, etc.). These values have been chosen so that the associated communication rates range from 100% to 5% in small equal increments. Therefore, the associated set of CR values is

$$CR_{all} = \{100, 90, 80, 70, 60, 50, 45, 35, 25, 15, 5\}$$

The communication rate is expressed in percent.

For the communication success probability, the values are

$$CP_{all} = \{100, 80, 60, 40, 20\}$$

which are also expressed in percent.

The vehicle motion is determined by the control input that is applied to the agents in the network. We study 4 main types of motion:

$$MT_{all} = \begin{matrix} \{\text{circular concentric}, \\ \text{slow circular non-concentric}, \\ \text{fast circular non-concentric}, \\ \text{fast non-circular}\} \end{matrix}$$

Finally, we study 4 main classes of communication graphs. The first type is a fully connected, 2-agent graph, where both agents measure and communicate with each other. This allows us to

www.manaraa.com

analyze the performance of our event-based filter in a simpler scenario, where network effects do not play a big role. Then, the effects of a more complex network topology can be separated in subsequent simulations, where more agents and other ways to communicate are introduced. In order to do that, a bigger network of 6 agents is analyzed, with 3 different graphs: a star graph, where one agent (called central agent) is connected to all other agents, and all other agents are only connected to the central agent; a bridge graph, with 2 cliques of 3 agents each that are fully connected, and with just one link between the cliques; and a chain (or line) graph, where each agent is only connected to the next and previous agents. The first graph is supposed to be a realistic baseline we can compare the rest of results with. This configuration, with one agent acting as a hub, is safe, does not incur on high communication costs, and can be used in a variety of real-life applications. The next graph we study, with 2 cliques and 1 bridge, has the additional complication that one of the cliques is more links away to the agent with accurate positioning; we expect to see different behaviors between the 2 cliques under some circumstances. The line graph is yet more challenging, in the sense that *any* communication failure will render one entire group of agents blind to accurate information.

A total of $n = 30$ i.i.d. runs are performed for each single parameter combination.[3] To generate all parameter combinations, all parameters but one are fixed, and the one that is not adopts all the values previously determined. Once all values have been used for the first parameter, the second parameter will be changed to the next value, and the first parameter will again adopt all possible values. This process will be repeated until all possible combinations have been realized. If we have v_{p_i} possible values for parameter i, then the total number of combinations is

$$n_{total} = \prod_{i=1}^{n_p} v_{p_i}$$

where n_p is the number of different parameters considered, in our case 4.

General parameter values for the simulations, both with 2 agents and 6 agents, can be found

[3] This number is generally considered the smallest one to use for the law of large numbers to apply, and due to limited computer time is the one we will use, since we are mainly trying to obtain the mean values of different magnitudes.

www.manaraa.com

in the following tables.

Table 5.1: Table with the values of the parameters that are common for all simulations.

Parameter	**Value**
Duration of simulations, T	10s
Time step, dt	0.1s
Process noise covariance matrix, Q	$\begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.001 \end{bmatrix}$
Range error variance, ρ_r	0.05m^2
Bearing error variance, ρ_b	0.05rad^2
GPS-position variance, $\rho_{GPS_{pos}}$	1m^2
GPS-heading variance, $\rho_{GPS_{hd}}$	1rad^2

Table 5.2: Table with the values of the parameters that are common only for the 2-agent simulations.

Parameter	**Value**
Initial state, $\hat{X}_0^{\dagger}$	$\begin{bmatrix} -2\text{m} & 0 \\ 12\text{m} & 5 \\ 2\pi/3\text{rad} & -\pi/2 \end{bmatrix}$
Initial covariance, $P_0^{\ddagger}$	$\mathbb{I}_{3\times 3}$
Control input, $u(k)^*$	Motion 1: $\begin{bmatrix} 1\text{m/s} & 1 \\ 1\text{rad/s} & 0.5 \end{bmatrix}$ Motion 2: $\begin{bmatrix} 2\text{m/s} & 2 \\ 1\text{rad/s} & 1 \end{bmatrix}$ Motion 3: $\begin{bmatrix} 1\text{m/s} & 0.5 \\ 1\text{rad/s} & 0.5 \end{bmatrix}$ Motion 4: $\begin{bmatrix} 1\text{m/s} & 1 \\ \sin(0.5\,t(k)+\pi)\text{rad/s} & \sin(0.1\,t(k)) \end{bmatrix}$

† Each column represents the state of an agent.
‡ Covariance matrix associated to each state.
* Each column represents the control input for an agent.

www.manaraa.com

Table 5.3: Table with the values of the parameters that are common only for the 6-agent simulations.

Parameter	**Value**
Initial state, $\hat{X}_0^{\dagger}$	$\begin{bmatrix} 0\text{m} & -5 & 5 & 5 & -5 & 0 \\ 0\text{m} & 7 & 12 & -12 & -7 & 17 \\ 0\text{rad} & \pi/2 & \pi/2 & 0 & 0 & -\pi/2 \end{bmatrix}$
Initial covariance, $P_0^{\ddagger}$	$\begin{bmatrix} 1\text{m}^2 & 0 & 0 \\ 0 & 1\text{m}^2 & 0 \\ 0 & 0 & 0.1\text{rad}^2 \end{bmatrix}$
Control input, $u(k)^*$	$\begin{bmatrix} 1\text{m/s} & 0.5 & 1 & 0.5 & 0.7 & 0.5 \\ 0.5\text{rad/s} & 1 & 1 & 0.5 & 0.1 & 0.5 \end{bmatrix}$

† Each column represents the state of an agent.
‡ Covariance matrix associated to each state.
* Each column represents the control input for an agent.

www.manaraa.com

Chapter 6

Results

In this section, results from all simulation runs are presented and discussed. Results are first divided into two subsections that, even though present a lot of similarities, differ in the level of complexity and number of phenomena involved. In the 2-agent simulations, inherent characteristics of the event-based algorithm manifest in a more isolated way, therefore being easier to analyze these cases and draw conclusions. Conversely, 6-agent simulations have an added layer of complexity, since we are introducing network topology effects, non-ubiquitous GPS measurements and covariance intersection. By first outlining and discussing 2-agent simulations and subsequently transitioning to 6-agent simulations, we hope to introduce the results in an order that makes sense and that is as clear as possible.

6.1 2-agent simulations

We use simulations with $N = 2$ to study the effects of the following parameters: 1) the innovation threshold, indicated by δ, which directly correlates with a more intuitive metric that is the communication rate, 2) the communication success probability, indicated by CP, and 3) the type of motion that the vehicles follow, indicated by MT. The parameters used in the simulations for 2 agents are shown in Table 5.2, Chapter 5.

www.manaraa.com

6.1.1 Effects of innovation threshold

When studying algorithms whose main goal is to help reduce communication costs in a robot network, a straightforward metric is the amount of data that is taken, processed and shared between agents.

In this section we show the effects on filter performance and consistency of the innovation threshold, δ for perfect communication scenarios, that is, when the communication success probability CP is 1.

In Figure 6.2, we can see one of the most important characteristics of this filter, which is its ability to maintain consistency in perfect communication scenarios. This figure shows the result of agent 1 tracking both its own and agent 2's pose in a 2-agent setup, where there is no apparent loss of consistency as we increase δ (thus decreasing the average explicit communication rate) because the mean squared error matches the predicted covariance showing that even with the nonlinear dynamics and measurement models, our algorithm remains consistent. For reference, we can look at Figure 6.1 to see that, for the largest value of the innovation threshold considered, $\delta = 1.5$, the communication rate is $CR(\delta = 1.5) = 9\%$, and a communication rate of 50% corresponds to $\delta = 0.3$.

We also compare the MSE of our event-based algorithm (referred to as full EBF hereafter) with an event-based filter where implicit measurements are *not* fused (referred to as EBFni from now on) in Figure 6.3 for a 2-agent configuration. The two filters are set up to run in parallel with the same data and initial conditions. This means the conditions to decide whether to send a measurement or not were computed only once and provided the same trigger for the two filters; in the event that the measurement was to be sent (shared explicitly), both filters performed the traditional Kalman update, but if the measurement was to not be sent (shared implicitly), then the full filter performed an implicit update and the filter with no implicit information did nothing.

We can see that, as δ increases and fewer measurements are sent, our algorithm steadily outperforms the version without the implicit information. This clearly shows that fusing implicit

www.manaraa.com

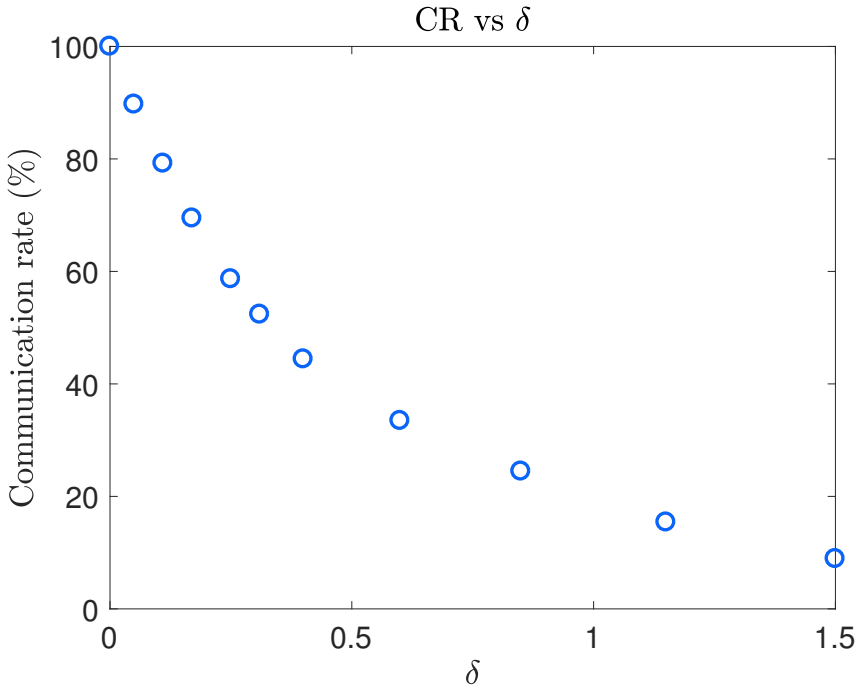
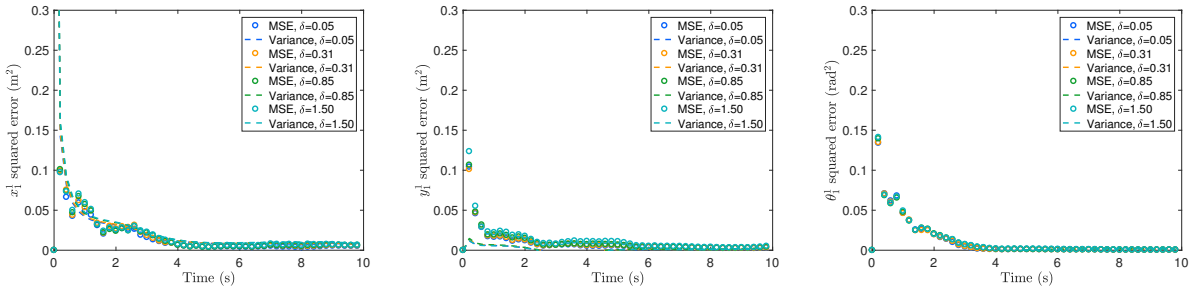


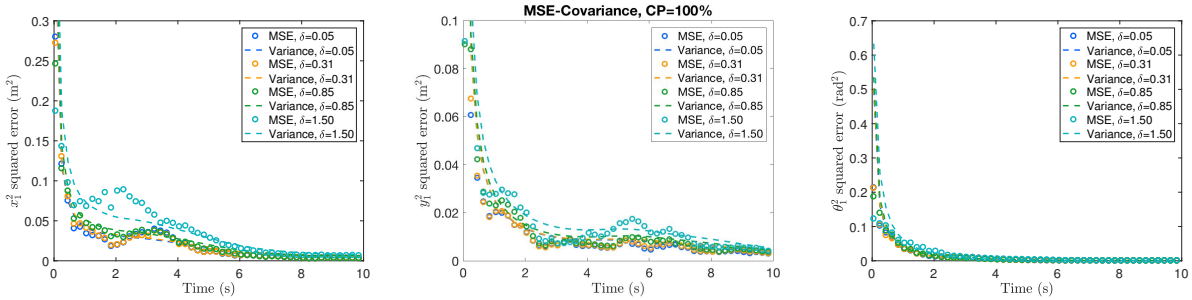
Figure 6.1: This figure shows the averaged communication rate between the 2 agents as a function of the innovation threshold, δ.

information in the measurements not sent improves performance. Furthermore, for small δ values, both filter's performances will be very close to an EKF that fuses all measurements explicitly. This is a point worth underlining – for the scenarios tried in the 2-agent simulations, the performance of the full event-based filter is barely worse than that of a centralized EKF for communication rates as low as 50%, both in terms of MSE and of the matching between MSE and covariance. For larger δ values, our algorithm has moderate increases in MSE for the benefit of requiring much fewer measurements sent, and manages to keep up with the EKF until the communication rate drops below 10%, where the performance drops significantly. Another characteristic observed that is desirable is the fact that performance degrades gracefully, keeping consistency throughout most of the innovation threshold spectrum.

www.manaraa.com



(a) Agent 1 tracking its own pose.



(b) Agent 1 tracking agent 2's pose.

Figure 6.2: This figure illustrates that there is no apparent loss of consistency as we increase δ. For both agents' states, and for all components, the mean squared error matches the predicted covariance.

6.1.2 Effects of communication success probability

The event-based algorithm fuses information into its state estimates in two main ways (three if we include covariance intersection, although this method does not fuse measurements but states and covariances and is not implemented in the 2-agents simulations): explicit measurements are used to update the state estimate using a traditional Kalman filter, and implicit measurements are fused using the implicit algorithm defined in previous sections, which in turn uses the knowledge about the innovation threshold. A potential source of problems is, therefore, the loss of data packets through the communication channel, since the algorithm relies on the knowledge that a measurement which was not received was deliberately censored. Whenever this happens, the receiving agent will process the lost measurement component implicitly, which may or may not be the case. Here, we investigate the robustness of our algorithm to imperfect communication

www.manaraa.com

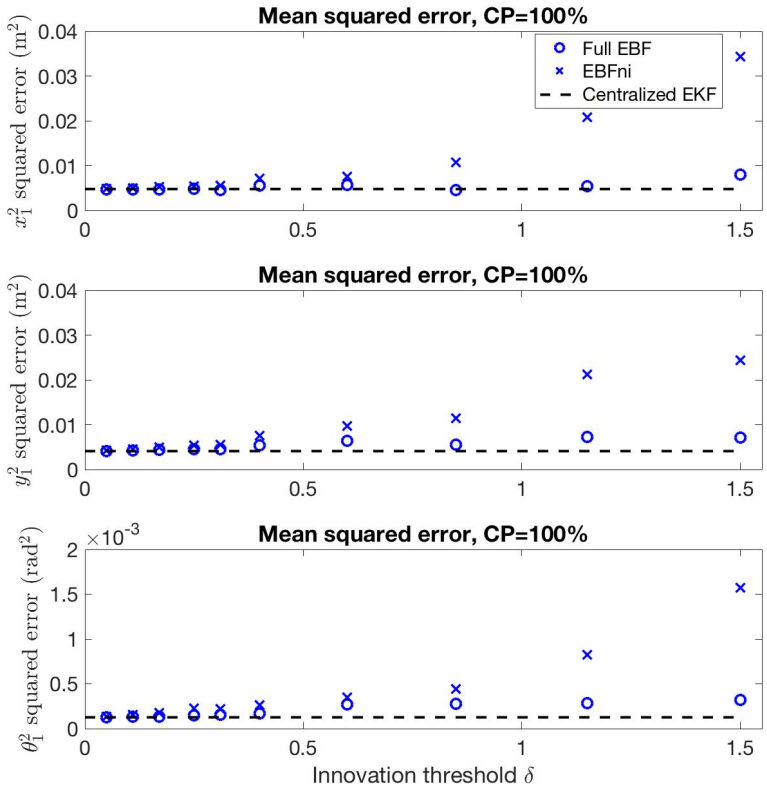


Figure 6.3: This figure depicts our event-based filter against an event-based filter that does not fuse negative information. One can see that, as the threshold parameter δ increases, our filter steadily explicitly sends fewer measurements while not increasing squared error much above the full EKF ($\delta = 0$)

channels, specifically looking at filter accuracy and consistency for different communication success probabilities.

Figure 6.4 shows the performance of two agents in a case where 50% measurements are being explicitly sent on average, and different percentages of those sent are dropped due to imperfections in the communication channel. This percentage is what we refer to as the communication success probability, or just communication probability, CP. 6.4 (a) shows agent 1's estimate of its own x coordinate, x_1^1. Since absolute pose measurements are readily available at every time step for both agents, the estimates of the agents' own states are not expected to be sensitive at all on CP, which is seen in the plot. However, 6.4 (b) and (c), show that agent 1's estimate of agent 2's x location, x_1^2,

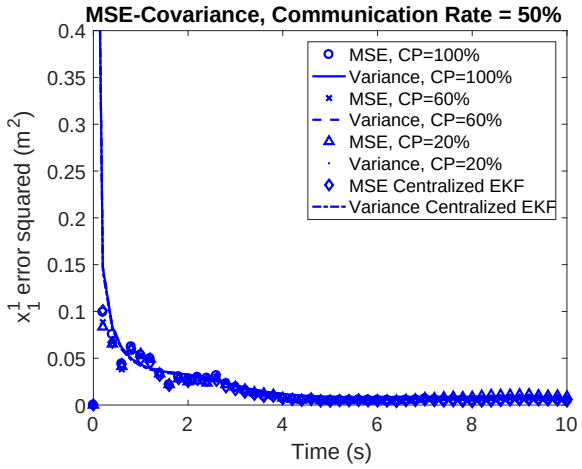
www.manaraa.com

as well as agent 2's estimate of agent 1's x location, x_2^1, are both dependent on CP. One may argue that neither agent actually *needs* the other agent's proprioceptive measurements (obtained through GPS), since by getting proprioceptive measurements of itself (which provide information about the pose in an absolute reference frame) and relative measurements to the other agent, all states can be recovered. While that is true, let us keep in mind that the assumption is that measurements are being shared between team members. As such, a received measurement will be fused explicitly, and a non-received one will be fused implicitly, while ignoring measurements is not considered. By fusing implicit information, the algorithm assumes that the original measurement was *within* some bounds. At the same time, a measurement that has been sent is, by definition, *outside* of these bounds. The problem comes when a measurement is sent but not received, since then it is assume to contain some information when in reality it contains other information. Filter inconsistency is, therefore, an inevitable consequence when data packets are dropped, and we see that as we increase communication failure rates there is an increasing gap between the predicted covariance and the true MSE.

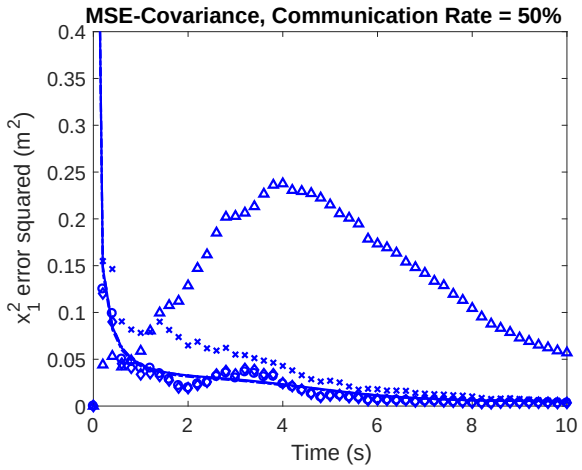
Figure 6.5 depicts the final sum of the MSE for both agents for several communication probabilities, showing that the error increases as more measurements are dropped. 6.6 shows the predicted trace of the covariance matrix. One thing to note is the difference in scale between the two plots, 6.5 and 6.6. If we focus on curves corresponding to large CP values, we see that the points match quite well, which at a first glance indicates a consistent filter. However, as we decrease CP we see what appears to be as the filter being overconfident in the filter's estimates. The lowest value of the communication probability shown in the MSE plots is 40%, since 20% are too large to fit in the frame.

One of the key factors causing this mismatch between predicted and actual errors is the filter interpreting dropped measurements as implicit measurements, which leads to incorrectly fusing observations. Only messages shared explicitly can be misinterpreted. That is, a message that has been sent but not received is an explicit message that has been fused implicitly, but a message that was intentionally shared implicitly will never be fused explicitly, since the value of the measurement

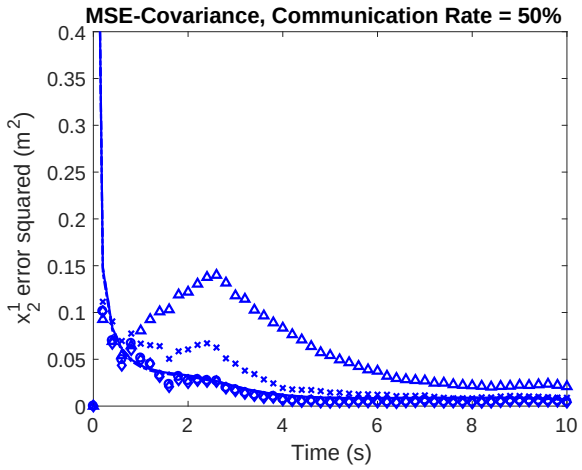
www.manaraa.com



(a) Agent 1's estimate of its own x coordinate.



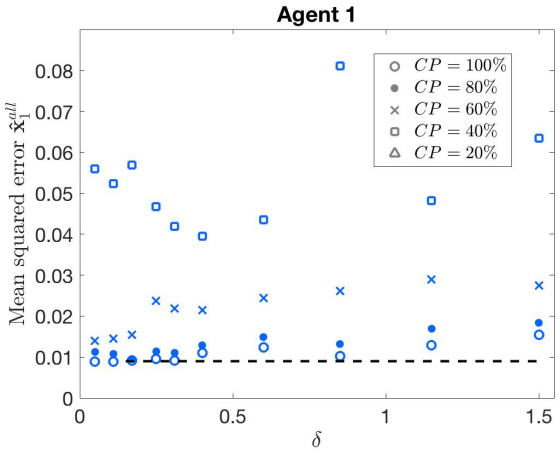
(b) Agent 1's estimate of agent 2's x coordinate.



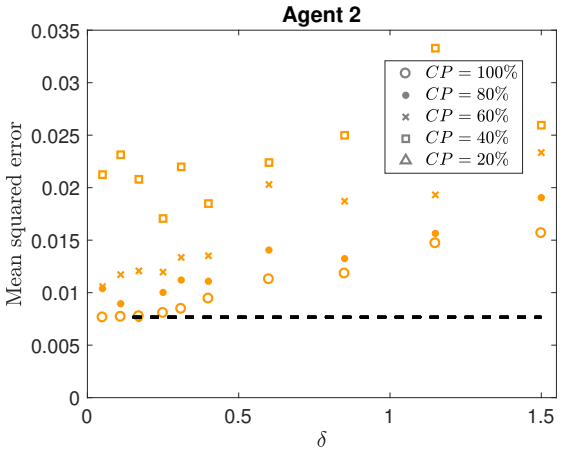
(c) Agent 2's estimate of agent 1's x coordinate.

Figure 6.4: Consistency loss for a scenario with 50% communication rate.

www.manaraa.com

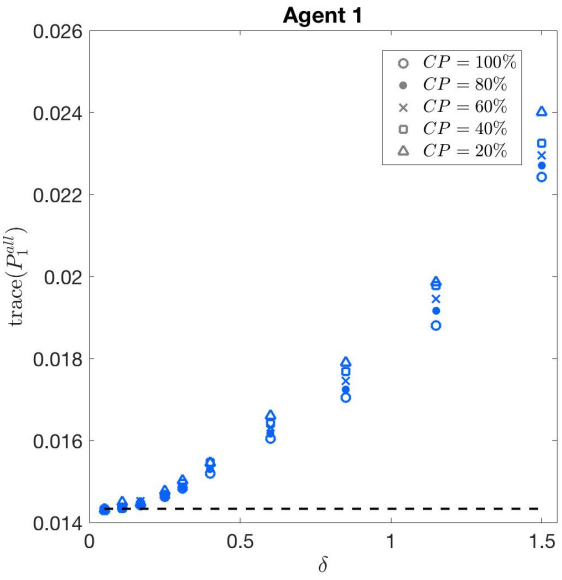


(a) Sum of the components of the mean squared error in agent 1's estimate, corresponding to agent 2's state.

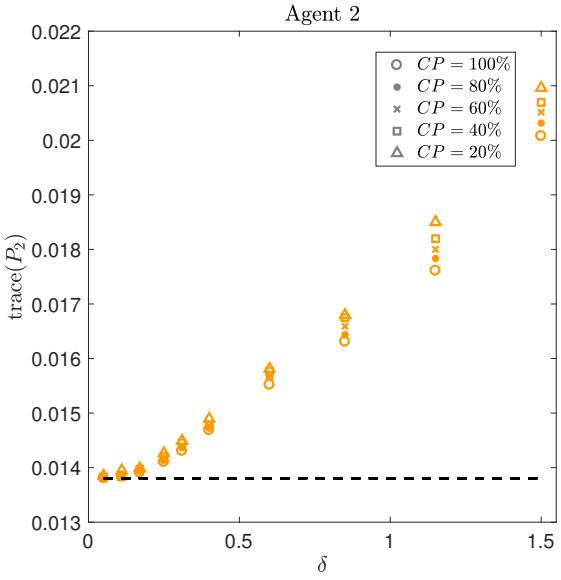


(b) Sum of the components of the mean squared error in agent 2's estimate, corresponding to agent 1's state.

Figure 6.5: Sum of the components of the mean squared error for different CP values, as a function of δ.



(a)



(b)

Figure 6.6: Trace of the covariance matrix associated with the state of the network in (a) agent 1 and (b) agent 2, at the final time step for different innovation threshold values.

will not be available to the receiving agent to begin with. As δ increases, more measurements are

www.manaraa.com

sent implicitly and fewer explicitly. Accordingly, fewer measurements can be dropped (in absolute terms. The percentage of dropped measurements is fixed), which results in fewer measurements being incorrectly fused.

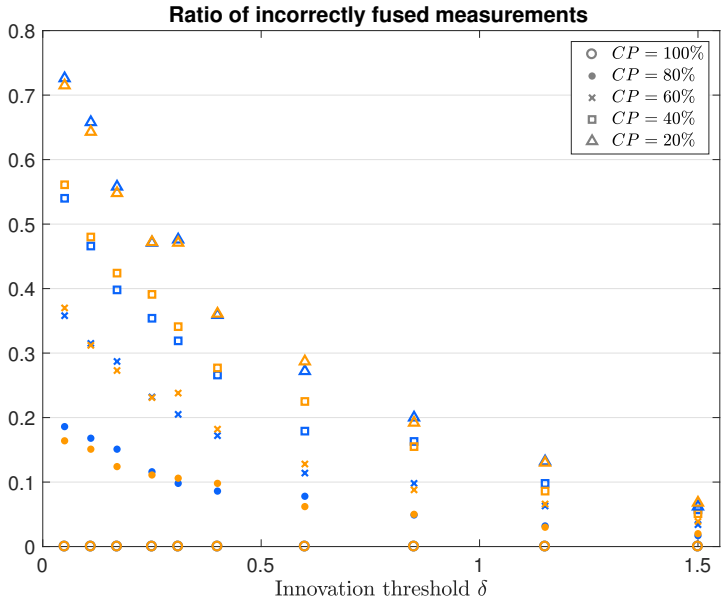


Figure 6.7: Ratio of incorrectly fused measurements over total shared measurements for different CP values as a function of δ. Agent 1 is shown in blue, and agent 2 in orange.

In order to explain numerically this particular behavior of the filter, we look at the relative amount of measurements of messages that are not fused correctly (by not fused correctly, we refer to messages that are fused as if they were implicit but that were actually explicit). This quantity, which we refer to as the incorrectly classified ratio (ICR), is equal to the number of measurements that a vehicle has incorrectly classified over the total number of measurements that the same vehicle has received from other agents and fused (Equation 6.1). Intuitively, the this ratio seems like a good candidate to explain the origin of the filter's overconfidence (the state error distribution does not agree with that predicted by the covariance matrix, and in particular errors are larger than the covariance matrix suggests). However, Figure 6.8 suggests that, even though up to a certain point, as the ratio of incorrectly fused measurements decreases, the actual and predicted errors get closer to one another, they start diverging again for larger innovation threshold values. This can be due

www.manaraa.com

to the fact that, even if for larger δ values this ratio is lower (as seen in Figure 6.7), the filter is more sensitive to an explicit measurement drop, since most of the measurements it is receiving are in the form of implicit information.

$$ICR_{ij} = \frac{\text{measurements dropped}}{\text{total measurements}} \tag{6.1}$$

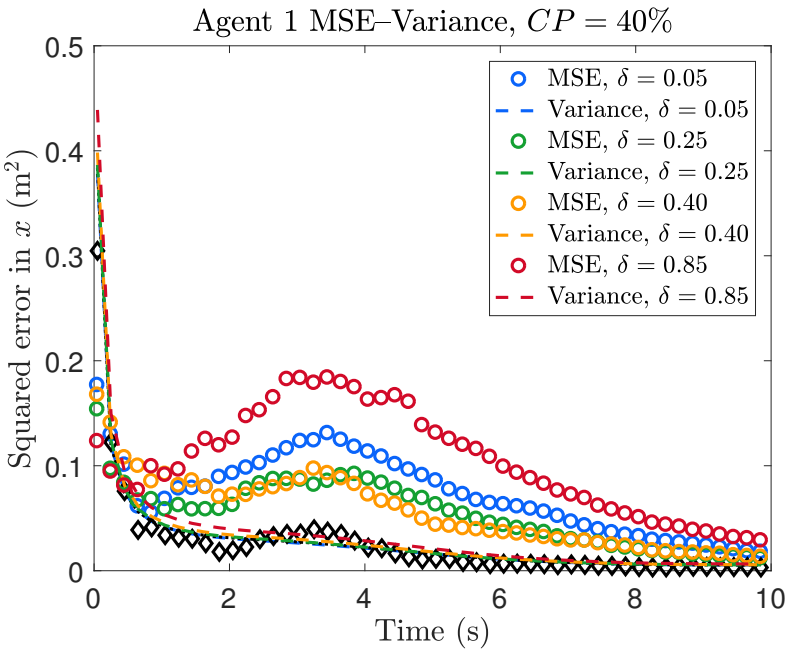
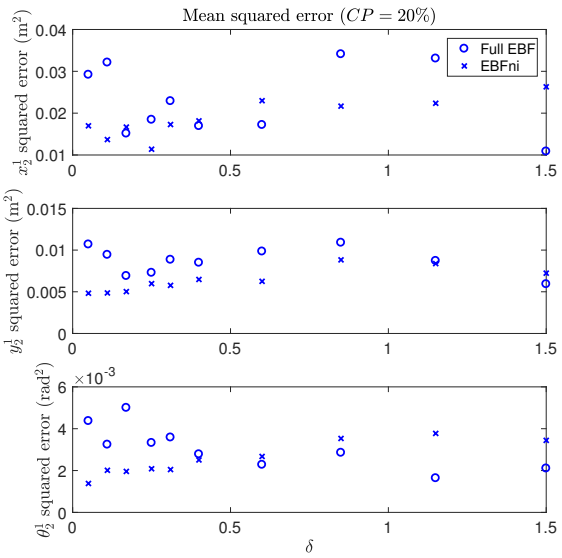


Figure 6.8: Comparison between MSE and variance for different δ values, for a communication probability of 40%. We can see that the smallest gap between MSE and variance is associated with intermediate innovation thresholds (green and orange). The diamond-shaped markers correspond to the baseline, the centralized EKF.

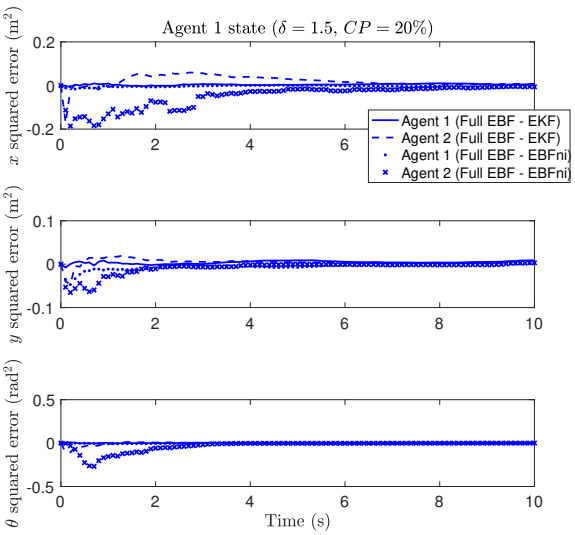
Figure 6.9 compares our event-based filter (EBF − EKF) with one where no information is fused when our filter would have sent an implicit measurement (Full EBF − EBFni) when the communication probability (CP) is low. This contrasts with Figure 6.3 that highlights the improved performance of fusing the implicit information when $CP = 100\%$. Figure 6.9 (a) and (b) both shows a limitations of our event-based filter. Here, the communication probability is only 20% and, accordingly, some of the messages are in the form of explicit measurements that are dropped, even though our filter interprets them as having been implicitly sent, as seen in Figure 6.7. This

www.manaraa.com

leads to larger errors in Full EBF − EKF and smaller errors when implicit information (real implicit information or dropped measurements) is ignored (Full EBF − EBFni). 6.9 (a) also implies that for low communication probability, increasing the innovation parameter δ improves the performance. This seems to be explained by the fact that increasing δ decreases the chance that an explicit measurement is dropped, and therefore, misinterpreted as an implicit measurement, as seen above in 6.7.



(a) Mean squared errors for the two filters, the full EBF and the EBF with no implicit information, as a function of δ.



(b) Mean squared errors for the two EBF filters and the centralized EKF, as a function of time.

Figure 6.9: Comparison between the mean squared errors of the two event-based filters analyzed.

6.1.3 Effects of vehicle motion

Intuitively, we would expect the states of agents that are moving following more aggressive maneuvering, including sharp turns and larger acceleration values, to be harder to estimate for the event-based filter. It is important to remember that the EBF is based on the extended Kalman filter, which is in turn not guaranteed to work in non-linear cases and can even diverge. By introducing the implicit update algorithm in our code, the filter becomes more vulnerable to nonlinearities. Here,

www.manaraa.com

4 different types of vehicle motion with varying degrees of non-linearity are considered; circular concentric, slow circular non-concentric, fast circular non-concentric and aggressive maneuvering, as can be seen in Figure 6.10.

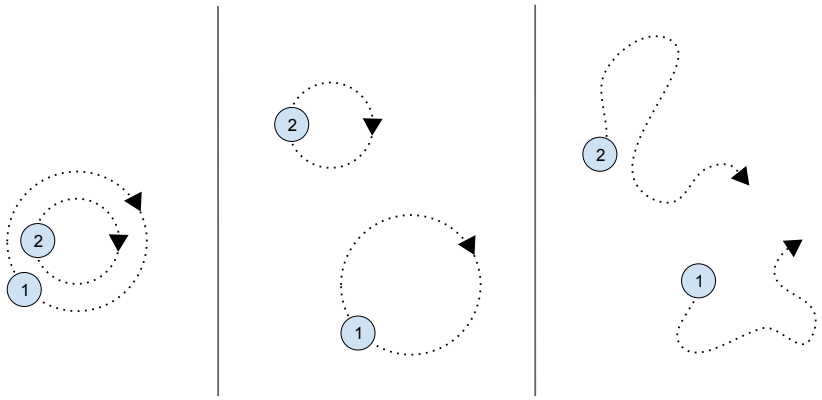


Figure 6.10: The three main motion types analyzed. For the circular concentric case (shown here in the middle), two different configurations exist: one with low velocities and one with high velocities.

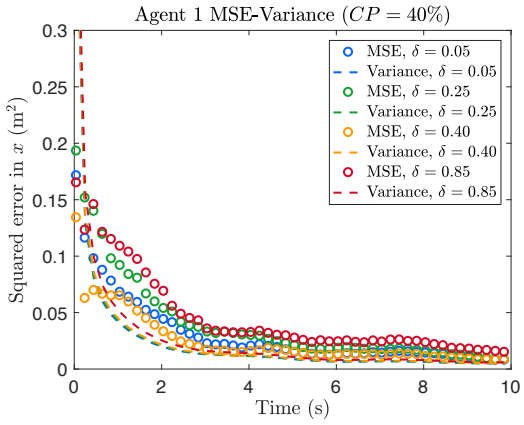
6.2 6-agent simulations

In this section, we perform simulations with a larger network where $N = 6$ in order to study the performance of the event-based filter for a larger network with different communication topology models. Since our ultimate goal is to see the accuracy and consistency of this algorithm in scenarios which are as realistic as possible, here we get one step closer to that by introducing multiple agents, different communication graphs and non-ubiquitous GPS in the simulations. Figure 6.12 depicts the 6-agent simulations for the three different graphs or topologies: star, bridge and line. The parameters used in the simulations for 2 agents are shown in Table 5.3, Chapter 5.

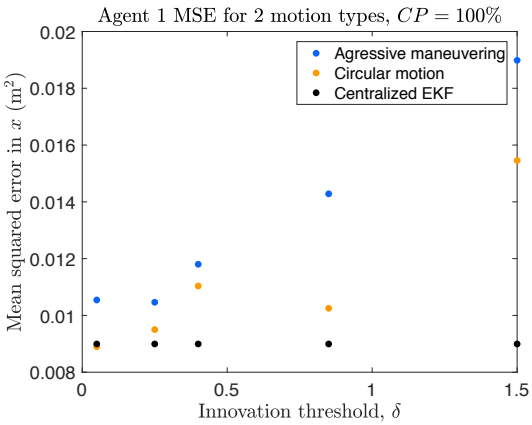
The most noteworthy characteristic we can infer from the results is that the motion type does not seem to cause any significant differences in the general behavior of the filter. Even though the specific numbers for different metric are slightly different and in general slower motions show lower MSE values, the same trends are observed across the different motions. This explains the fact that we have chosen that all figures in previous sections only depict the motion type in which

www.manaraa.com

the agents move more aggressively. This gives us the confidence that, when showing any specific effect and quantifying it, we know it will also hold, and most likely to a lesser extent, for other motion types. In Figure 6.11 (a) we show that, for a communication probability of 40%, the filter performs significantly better when estimating vehicles that follow a slower, more uniform motion than when estimation vehicles that move aggressively. Figure 6.11 (b) measures the effects of the motion type on the MSE, and the MSE for the centralized EKF is included as a reference.



(a) Comparison between MSE and variance for different δ values, for a communication probability of 40%, for a circular non-concentric motion. This can be directly compared with Figure 6.8, which shows the same for the motion with aggressive maneuvering.



(b) Mean squared errors for two motion types (circular non-concentic, which shows the lower MSE, and aggressive maneuvering, which shows the higher MSE) and the centralized EKF, as a function of the innovation parameter.

Figure 6.11

6.2.1 Effects of communication graph

The simulations results show that communication topology plays an important role in the performance of the algorithm. In this section, the performance of the event-based filter for a larger network consisting of 6 agents is studied. Since our ultimate goal is to see the accuracy and consistency of this algorithm in scenarios which are as realistic as possible, additional elements such as multiple agents, different communication graphs and non-ubiquitous GPS are added in the

www.manaraa.com

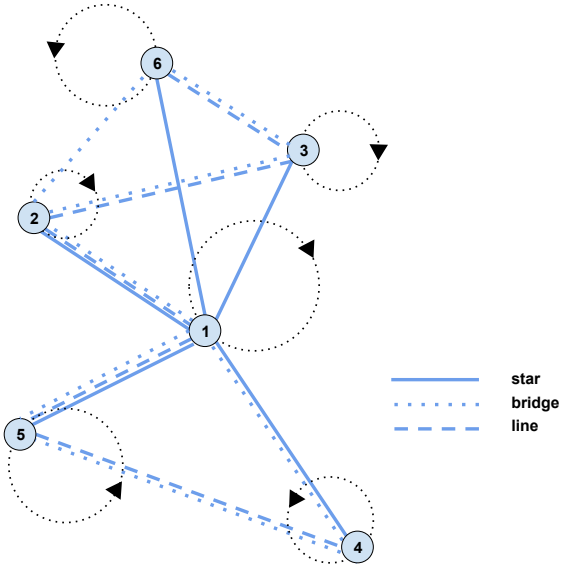


Figure 6.12: Vehicle motion and the 3 different communication models (star, bridge, chain) used in the 6-agent simulations.

simulations. Figure 6.12 depicts the 6-agent simulations for the three different graphs or topologies: star, bridge and line.

Two main aspects of the 6-agents simulations, observability and covariance intersection effects, are considered next: observability issues and covariance intersection effects.

1. Observability issues

Depending on the communication topology and GPS availability to the agents, the state of the whole network in an absolute reference frame may be impossible to determine. The ability to pin down a subset of agents does not necessarily lead to the ability to do so for all team members. In our algorithm, measurements are shared between communicating pairs of agents, but they are not passed over to additional members.

Observability of the full network state is heavily dependent on the measurement sharing topology. For instance, in the bridge topology the 2 subsets of agents have full communication internally (every agent talks with the rest of the agents in the subset), but there is only one link between the two groups. This means if that link fails or is lost, the two groups become blind to

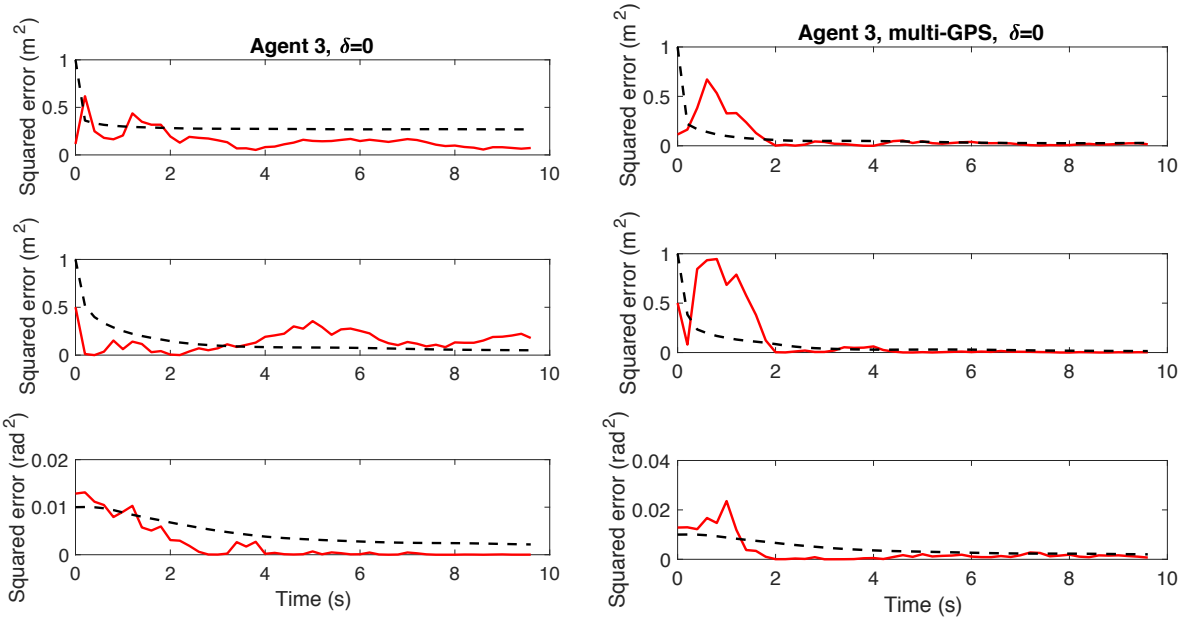
www.manaraa.com

one another. A more extreme case is the chain topology, where the network diameter is largest. For this graph, if information is desired to go from the agent at one edge to the agent at the other edge, all agents in between have to successfully receive and transmit that information at some point, which takes time and makes the system more vulnerable to data drops. An example of such valuable information that would potentially be shared among all agents is GPS localization, since the problem of interest is that of robot localization in a global reference frame. On the other side, the star graph has a smaller network diameter, which benefits the system as a whole, but is dangerous for agents individually, since if one of the links breaks, an agent becomes isolated. Complexity increases in situations where GPS is limited or restricted to certain agents and, overall, the particular application dictates which topology should be used, and different communication graphs work better in different scenarios.

There are different ways to cope with observability issues. Well connected networks present fewer problems when it comes to this thanks to more links existing between agents, at the expenses of having increased communication costs and, possibly, having to use higher bandwidths and more expensive hardware. Another option is to have GPS or other forms of absolute positioning available to more agents, or to agents in strategical positions of the network. As an example, let us examine a case with a chain-shaped graph. Here, it can be seen that even if the agent with GPS access changes, most agents are still going to be out of reach, not being able to benefit from that information. For the chain graph, Figure 6.13 (a) shows a case where only one of the agents (agent 4, in the edge of the graph) gets GPS measurements. Here, the y component of the estimated state drifts and the filter is unable to correct it if only one agent gets GPS measurements. In (b) this behavior is corrected by adding additional agents (agents 1 and 6) that receive GPS measurements as well.

However, in some cases, such as for unmanned underwater vehicles, this may be impractical, since vehicles have to surface to get absolute measurements, thus incurring in additional energy consumption and losing valuable mission time. In more extreme scenarios this practice is not even possible, depending on mission requirements, GPS signal availability or hardware used. The third method, which is studied in the present paper, is covariance intersection (or CI), and allows agent

www.manaraa.com



(a) (b)

Figure 6.13: Comparison of agent 3's resulting state estimates between (a) a case where just agent 4 has GPS, and (b) a case where multiple agents have GPS. Both plots correspond to the chain graph.

pairs to fuse their state estimates and covariances with the goal of reducing uncertainty and obtaining an estimate that outperforms the other two. In this way, accurate state estimates from an agent that benefits from absolute measurements flow through the network.

2. Effect of covariance intersection

Covariance intersection is used as a way for agents to sync their estimates and reduce their covariance matrices. It is worth noting that CI involves higher communication and computational costs than performing a traditional Kalman measurement update, so it is not used as the main method for data sharing and fusion, but rather to overcome the problems associated with less observable networks by triggering it when the trace of the covariance matrix for a particular robot exceeds a certain threshold. The value of this threshold is design choice in which different metrics and problem requirements have to be considered.

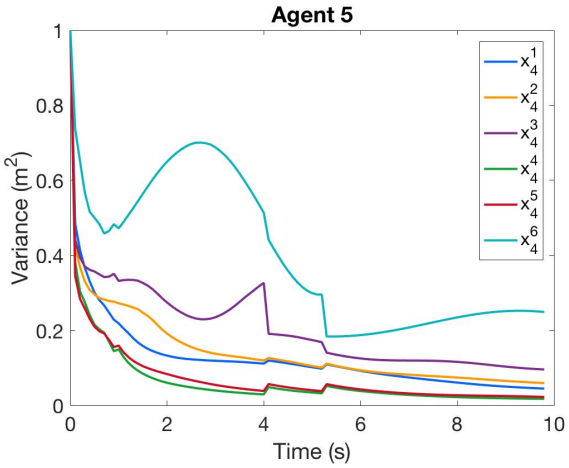
www.manaraa.com

Figure 6.14 shows the effect that performing CI has on diagonal elements of the covariance matrix of an arbitrary agent. The sinusoidal shape of the curves is caused by the circular motion of the robots. This figure depicts a simulation where the agents communicate with one another following a chain-shaped (or line) graph, as can be seen in Figure 6.12. By introducing additional correlations between agents' states, the resulting covariance matrix is generally filled with non-zero values in corresponding off-diagonal elements. CI effects manifest by generally sudden jumps in the covariance values, as can be seen around $t = 4$s or $t = 5.2$s. These correlations are passed from agent to agent every time that CI is performed. However, states corresponding to two agents that are separated by several links in the communication graph will not become correlated instantaneously. Instead, it will take intermediate agents to perform CI successively for a few time steps. Delays in the propagation of covariance intersection correlations are hard to see in these simulations, since the network is relatively small.

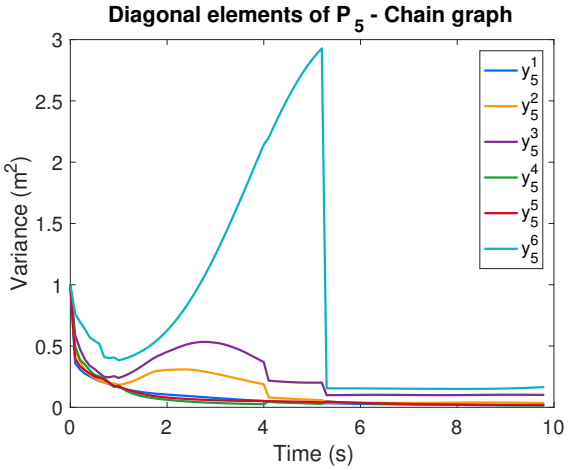
Another interesting aspect of these simulations is the fact that the covariance matrix hits an upper bound even in cases where there are very sparse or no measurements containing information about specific agents. Then, every time covariance intersection is performed new correlations between states are added, which reflects in this bound adopting a different value. Figure 6.14 shows the variances of the states of all agents, as estimated by agent 5. As can be seen, there is a direct correlation between the number of links separating agent 5 and the other agents and how large the variances are – for example, agent 6 is the farthest away from 5 (in terms of links or connections) and the associated variance for 6 is the largest, whereas agent 2 is measured by 5 directly, so its covariance is small.

Covariance intersection is not needed in networks where all agents' states are measured by or shared between one another, since if the filter is properly tuned it will eventually converge. Figure 6.15, where agents communicate following a star-shaped graph, shows that because the only agent that is receiving GPS measurements (agent 1) acts as a hub and is able to share them with the rest of the network, all other states can be uniquely estimated by virtue of pinning down agent 1. On the other hand, if GPS measurements were provided only to one of the agents that act as leaf

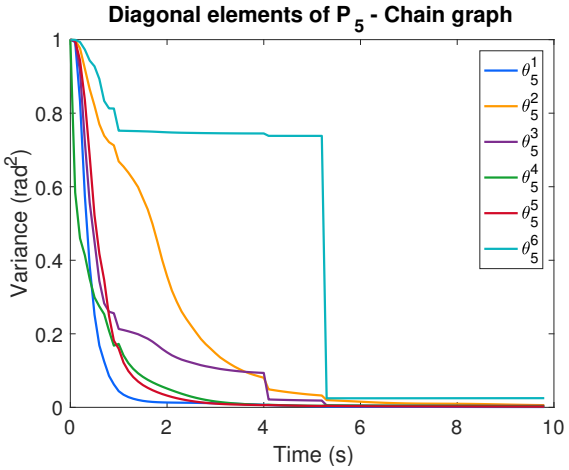
www.manaraa.com



(a)



(b)



(c)

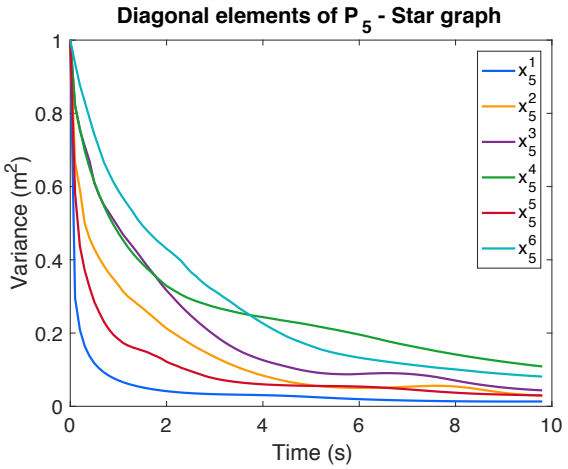
Figure 6.14: Sample simulation results showing component-wise variances as predicted by agent 5 in the chain graph. Sudden drops or increases are an effect of CI, a method that acts instantly on states and covariance matrices.

www.manaraa.com

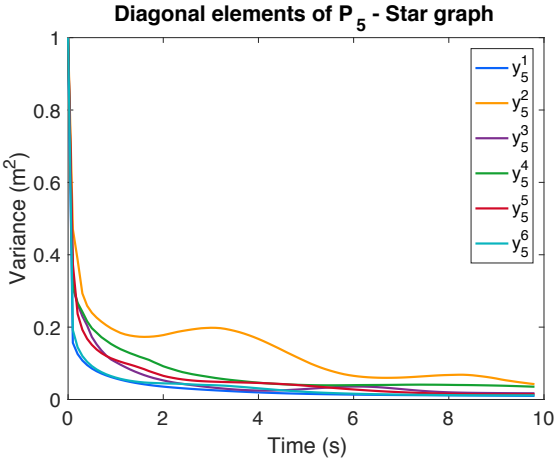
nodes (that is, all agents except for agent 1), it would become necessary to perform CI to prevent the estimates from drifting away from the true states.

One important conclusion in multiple agent scenarios is that the number of agents that have access to GPS measurements, as well as their position and ability to communicate with other agents, affects the overall performance of the filter. Additionally, in view of the results it becomes clear that CI plays a dominant role in cooperative localization, and correlations between agents states introduced early on in the simulations have a long-lasting effect that allows a reduction in communication costs without strong penalties on filter performance. This is particularly useful in poorly connected graphs, although it comes at a price – performing CI more often brings about higher communication costs, which is counterproductive, so an optimal combination of these two values needs to be obtained. This is a problem of its own that would be worth studying in future works.

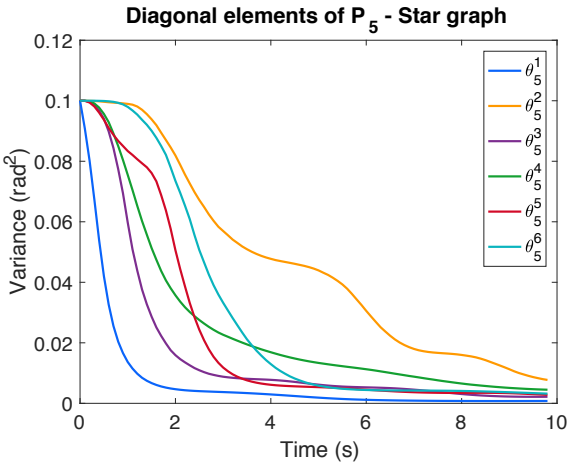
www.manaraa.com



(a)



(b)



(c)

Figure 6.15: Sample simulation results showing component-wise variances as predicted by agent 5 in the star graph. With the same CI threshold as in the chain graph, the better connected graph results in CI not triggering.

www.manaraa.com

Chapter 7

Conclusions

7.1 Summary of contributions

A novel algorithm that implements an event-based approach in a Kalman filter framework for cooperative localization problems has been proposed, described and studied. The algorithm does not make many of the assumptions that other works have made in the past and that, according to the author, greatly limit their applicability to actual robotic platforms. The method at hand offers vehicles that cooperate to localize themselves in an environment the possibility to check the value of information before sending it to other agents. This is made possible by adapting an event-based estimation approach to a decentralized cooperative localization problem, where each one of the vehicles does the sending, broadcasting and the fusion processes at the same time.

Prior work has been done with event-based estimation algorithms on simple systems where agents do not move at all, or move following very simple trajectories with linear dynamics models. The poses of the agents, which are the quantities to estimate, are observed directly too. In these situations, it is known that a Kalman filter is the optimal minimum variance estimator, and there are guarantees that it will work. However, real applications are not as simple as the models in these previous studies, and the performance of event-based filters remains unknown, for the most part. This work addresses some of the fundamental questions in the topic and constitutes a first step towards assessing the usefulness of event-based algorithms in the context of nonlinear cooperative localization.

Simulations have been performed for a variety of parameter values and scenarios to study the

www.manaraa.com

most significant aspects of the proposed method. One of these aspects is the ability to maintain consistency. Results show that the amount of data that is sent between communicating agents does not affect consistency in perfect communications scenarios; even in the cases where the communication rate, which directly measures what percentage of observations are shared explicitly, is below 10%, the updates do not result in optimistic or pessimistic error predictions, and the actual and predicted errors match well. The algorithm's MSE is compared to that of a centralized EKF, and it is found that communication rate can be decreased significantly before the MSE's for the two filters diverge significantly.

Imperfect communications are considered. Based on the results, the effects of inadvertently dropping data packets are tightly coupled with those of the communication rate. Specifically, while on high values of the communication rate the system is particularly sensitive to data losses, the extent of this is minimized as the communication rate decreases. This is partially explained by the rate of incorrectly fused data packets, which is much higher at high communication rates when a vehicle expects most of the data packets to be shared explicitly. A brief discussion on how to counteract these negative effects is included.

To the best of our knowledge, network topology effects have typically not been considered previously for decentralized event-triggered algorithms. In this work, 2 different network sizes are considered, one with 2 agents and another one with 6 agents. In the latter case, different communication models are studied and it is seen that, in situations where only one of the agents has access to absolute positioning information, the connectivity of this agent is crucial for the rest of the network to immediately benefit from that accurate information. Additionally, the way the network is globally structured also affects the rate at which privileged measurements reach all agents. It is seen that covariance intersection, a conservative fusion algorithm, can help introduce correlations between agents that increase the rate at which accurate information spreads. Covariance intersection can also be used sporadically as a tool to keep these correlations large enough so that the covariance matrices of agents remain bounded.

Finally, vehicle motion is also studied. Different types of nonlinear motion, with different

www.manaraa.com

speeds and turning rates, are looked into. Although the algorithm performs quantitatively worse in the cases where nonlinearities are larger, the overall effects and tendencies are the same, which is a very positive feature. Although no guarantees can be given as to the proper functioning of the algorithm in *every* situation, since it is based on an approximate solution, the different scenarios simulated here suggest that the algorithm is robust enough for a lot of practical situations.

One of the important contributions of this work is the fact that extensive Monte Carlo runs are performed to back the results with enough and meaningful data. As many technical details as possible (always advised by common sense and the precaution of not making this document too lengthy) are given so that other groups or individuals can reproduce the results obtained here.

On the whole, the event-based cooperative localization algorithm has proved to significantly reduce communication costs while compromising little on estimation performance. The performance degrades gracefully, as long as the linearity assumptions hold. Lossy communication scenarios should be handled with caution, and from an engineering perspective this should be one of the foremost issues to address in future works. In large networks, the presence of an algorithm such as covariance intersection is vital to keep the correlations of the agents large enough that the whole network benefits from access by one of the agents to accurate absolute positioning data.

7.2 Future work

Further immediate work could be done on changing or limiting the measurement types that vehicles are using in the current model to study the performance of the algorithm in yet more challenging situations. For example, making the GPS measurements more sporadic, or using only range instead of range and bearing. Another interesting issue would be to more formally quantify the communication and processing costs associated with running the algorithm, since the ultimate goal is to develop a method that is energy-efficient in real robot teams. Finally, variants of the current covariance intersection implementation promise to be more efficient computationally and have almost no differences in terms of result, for example by only fusing subsets of interest of the state vector [29].

www.manaraa.com

The proposed event-based algorithm attempts to primarily reduce communication costs by not communicating information indiscriminately, regardless of its predicted value. This makes the algorithm more computationally complex in certain aspects; the need to keep and update multiple estimates for communicating agents, to compute predicted innovations with the shared estimates or to perform the implicit update per se, which is more costly than a regular EKF measurement update, supposes an added layer of complexity in the algorithm. We found that, in a lot of scenarios, the communication needs are quite more stringent for the hardware platform being used than the processing ones. For example, in underwater scenarios data is usually sent acoustically, which requires an amount of power at least an order of magnitude higher than that associated with computations. A more formal metric to quantify this tradeoff is nonetheless recommended in future works. Additionally, throughout this work, communication rate is used as a measure of communication costs. The next natural step would be to be more specific in quantifying these costs. Knowing that the measurements vehicles take are in double, we can translate the bytes required to send these doubles to expected bandwidth.

The innovation threshold correlates to the communication costs directly, and is here defined by a value. From an engineering perspective, these values do not convey us much information about the communication costs. A more intuitive way to define the innovation threshold would be in terms of standard deviations, using our knowledge of the distribution of the innovation.

In the current model, vehicles keep an estimate of the states of *all* agents in the network. This was added in the event that the communications topology changes mid-simulation, in an ad hoc manner, and vehicles communicated with whatever other vehicles they ran into. A better way to deal with this situation would be that the vehicles exchanged states when they first encountered another vehicle they know nothing about, instead of keeping an estimate that does not contain any information, neither from direct measurements nor from measurements received from other agents.

Another possibility is exploring the robustness of the proposed algorithm to different sources of imperfections. One might be model errors; we have assumed some levels of process noise added to a perfect model, but in a lot of cases that does not hold, and the dynamics or control models have

www.manaraa.com

errors that could compromise the quality of the estimates. Another potential source of problems is robot failure or kidnapping; how is the system affected by the sudden loss of a robot? How many can be lost before the system is seriously compromised? Can the system recover if a robot is suddenly placed somewhere else and does not know it?

Requiring more thought and study, one relevant issue to address is finding a method for agents to handle the communication failures that ultimately lead to filter inconsistencies. It has been seen that the effect these inconsistencies play on the estimates depends upon other parameters, but in the worst cases it can lead to severe malfunctioning and underperformance of the filter. An idea is that agents take into account that it is unlikely to have a packet be completely censored by the event-triggering criterion. Then, an empty packet is almost certainly due to communication losses, rather than it not being sent intentionally. Another possibility is to exploit the knowledge of the vehicles' motions. For example, if for two agents, their heading angles do not change (we can imagine that they are moving in a straight fashion but in different directions), then the range measurements would be more innovative, since they contain information about the x and y coordinates of both agents, which do change, but the bearing measurements would not, since the angles between are a stationary quantity.

In the current work, the innovation parameter is user-fixed. In reality, predicting the system's response to a specific value of this parameter can be hard. We propose using heuristics to determine the correlation between the innovation parameter and the communication rates, since it depends of the system's properties such as dynamics, sensor specifications and environment. However, a significantly better solution would be for the system to adapt the value of the innovation parameter on the go, possibly using a reinforcement learning framework. In this way, based on the user's desired performance and energy consumption, the system would find an optimal value for the parameter based on rewards. Different modes could be used, for example, where these modes designate a "high-energy" or "low-energy" configuration.

www.manaraa.com

Bibliography

[1] S. J. Julier and J. K. Uhlmann, "Using covariance intersection for slam," Robotics and Autonomous Systems, vol. 55, no. 1, pp. 3–20, 2007.

[2] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," Proceedings of the IEEE, vol. 97, no. 2, pp. 427–450, 2009.

[3] A. J. Barry and R. Tedrake, "Pushbroom stereo for high-speed navigation in cluttered environments," in 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015.

[4] J. Sandee, W. Heemels, and P. Van Den Bosch, "Case studies in event-driven control," Lecture Notes in Computer Science, vol. 4416, p. 762, 2007.

[5] P. Swerling, "First-order error propagation in a stagewise smoothing procedure for satellite observations," 1959.

[6] R. E. Kalman et al., "A new approach to linear filtering and prediction problems," Journal of basic Engineering, vol. 82, no. 1, pp. 35–45, 1960.

[7] R. Kurazume, S. Nagata, and S. Hirose, "Cooperative positioning with multiple robots," in Robotics and Automation, 1994. Proceedings., 1994 IEEE International Conference on, pp. 1250–1257, IEEE, 1994.

[8] R. Kurazume, S. Hirose, S. Nagata, and N. Sashida, "Study on cooperative positioning system (basic principle and measurement experiment)," in Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on, vol. 2, pp. 1421–1426, IEEE, 1996.

[9] R. Kurazume and S. Hirose, "Study on cooperative positioning system: optimum moving strategies for cps-iii," in Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on, vol. 4, pp. 2896–2903, IEEE, 1998.

[10] I. Nourbakhsh, R. Powers, and S. Birchfield, "Dervish an office-navigating robot," AI magazine, vol. 16, no. 2, p. 53, 1995.

[11] R. Simmons and S. Koenig, "Probabilistic robot navigation in partially observable environments," in IJCAI, vol. 95, pp. 1080–1087, 1995.

[12] A. R. Cassandra, L. P. Kaelbling, and J. A. Kurien, "Acting under uncertainty: Discrete bayesian models for mobile-robot navigation," in Intelligent Robots and Systems' 96, IROS 96, Proceedings of the 1996 IEEE/RSJ International Conference on, vol. 2, pp. 963–972, IEEE, 1996.

المنارة للاستشارات

www.manaraa.com

[13] W. Burgard, D. Fox, D. Hennig, and T. Schmidt, "Estimating the absolute position of a mobile robot using position probability grids," in Proceedings of the national conference on artificial intelligence, pp. 896–901, 1996.

[14] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," Autonomous robots, vol. 8, no. 3, pp. 325–344, 2000.

[15] S. I. Roumeliotis and G. A. Bekey, "Distributed multi-robot localization," in Proceedings of the International Symposium on Distributed Autonomous Robotic Systems (DARS 2000), pp. 179–188, 2000.

[16] L. C. Carrillo-Arce, E. D. Nerurkar, J. L. Gordillo, and S. I. Roumeliotis, "Decentralized multi-robot cooperative localization using covariance intersection," in Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on, pp. 1412–1417, IEEE, 2013.

[17] S. J. Julier and J. K. Uhlmann, "A non-divergent estimation algorithm in the presence of unknown correlations," in American Control Conference, 1997. Proceedings of the 1997, vol. 4, pp. 2369–2373, IEEE, 1997.

[18] B. Noack, M. Baum, and U. D. Hanebeck, "Covariance intersection in nonlinear estimation based on pseudo gaussian densities," in Information Fusion (FUSION), 2011 Proceedings of the 14th International Conference on, pp. 1–8, IEEE, 2011.

[19] A. Prorok and A. Martinoli, "A reciprocal sampling algorithm for lightweight distributed multi-robot localization," in Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pp. 3241–3247, IEEE, 2011.

[20] J. Sijs and M. Lazar, "On event based state estimation.," in HSCC, pp. 336–350, Springer, 2009.

[21] J. Wu, Q.-S. Jia, K. H. Johansson, and L. Shi, "Event-based sensor data scheduling: Trade-off between communication rate and estimation quality," IEEE Transactions on automatic control, vol. 58, no. 4, pp. 1041–1046, 2013.

[22] D. Shi, T. Chen, and L. Shi, "An event-triggered approach to state estimation with multiple point-and set-valued measurements," Automatica, vol. 50, no. 6, pp. 1641–1648, 2014.

[23] S. Thrun, W. Burgard, and D. Fox, Probabilistic robotics. MIT press, 2005.

[24] S. S. Kia, S. Rounds, and S. Martinez, "Cooperative localization for mobile agents: A recursive decentralized algorithm based on kalman-filter decoupling," IEEE Control Systems, vol. 36, no. 2, pp. 86–101, 2016.

[25] S. E. Webster, R. M. Eustice, H. Singh, and L. L. Whitcomb, "Advances in single-beacon one-way-travel-time acoustic navigation for underwater vehicles," The International Journal of Robotics Research, vol. 31, no. 8, pp. 935–950, 2012.

[26] M. Ouimet, N. Ahmed, and S. Martínez, "Event-based cooperative localization using implicit and explicit measurements," in Multisensor Fusion and Integration for Intelligent Systems (MFI), 2015 IEEE International Conference on, pp. 246–251, IEEE, 2015.

www.manaraa.com

[27] S. Wilhelm et al., "Moments calculation for the doubly truncated multivariate normal density," arXiv preprint arXiv:1206.5387, 2012.

[28] G. M. Tallis, "The moment generating function of the truncated multi-normal distribution," Journal of the Royal Statistical Society. Series B (Methodological), pp. 223–229, 1961.

[29] S. J. Julier and J. K. Uhlmann, "Simultaneous localisation and map building using split covariance intersection," in Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on, vol. 3, pp. 1257–1262, IEEE, 2001.

[30] G. A. Bekey, Autonomous robots: from biological inspiration to implementation and control. MIT press, 2005.

[31] I. Rekleitis, G. Dudek, and E. Milios, "Multi-robot collaboration for robust exploration," Annals of Mathematics and Artificial Intelligence, vol. 31, no. 1-4, pp. 7–40, 2001.

www.manaraa.com