Decentralized and Event-Triggered Filtering for Position, Velocity, and Clock Bias Estimation

by

Dawson Edward Beatty

B.S., University of Colorado, 2019

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

2021

Committee Members:

Nisar Ahmed, Chair

Penina Axelrad

Morteza Lahijanian

Beatty, Dawson Edward (M.S., Aerospace Engineering)

Decentralized and Event-Triggered Filtering for Position, Velocity, and Clock Bias Estimation

Thesis directed by Assistant Prof. Nisar Ahmed

Teams of collaborative robots or rovers are expected to play a major role in future aerospace applications, including exploration of the lunar surface. Efficient, reliable methods to determine the position, velocity, and timing status of the rovers are required to facilitate this work and reduce risk of interference. This thesis presents two distributed Kalman Filter-based estimation algorithms for fully-connected networks using time-of-flight measurements. The network is composed of static beacons at known locations and mobile rovers whose positions and velocities are to be determined. Beacons and rovers are collectively called agents. Each agent's filter tracks the position, velocity, and time (PVT) states of all of the rovers in addition to the time states of the beacons. Knowledge of all position and velocity states enables rovers to accomplish collaborative tasks without interfering with each other; tracking time states supports the use of time-of-flight relative range measurements. This work addresses the limitations in the state of the art by introducing two decentralized algorithms where each agent can estimate the time-varying position, velocity, and clock bias/bias rate of itself and all other agents in a distributed fashion. In the first method, agents exchange full PVT state estimates, which are conservatively fused with local estimates using covariance intersection. In many applications it is desirable to reduce the amount of data transmission required for navigation purposes to free up space for other purposes. To that end, the second method reduces communication packet size by only sending sufficiently surprising measurements as determined by an event-triggering threshold. Simulation results using a low-quality clock with measurements precise on the order of centimeters show that the covariance intersection method achieves a 2D position RMSE error of 0.5 m. The event-triggering method significantly reduces required communication, but results in larger errors of about 1.8 m for a moderate triggering threshold of $\delta = 2\,\text{m}$.

Dedication

To myself in 2011, who was lonely, depressed, and nearly failing out of high school.

We made it.

Acknowledgements

First and foremost, thank you to Professor Nisar Ahmed for his guidance, advice, and endless enthusiasm for his field of work.

Thank you to Professor Penina Axelrad for repeatedly giving the opportunity to help out with her classes as a teaching assistant and being an excellent role model for the type of teacher I want to become.

Thank you to my best friend Annikka who makes working together in the office a downright joyful experience.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

Location awareness is crucial for many applications such as collaborative science exploration missions where rovers need to know their position and the position of other rovers as well. Time-of-flight measurements between two agents, converted to pseudorange by multiplying by the speed of light, can be used to determine the relative locations of the transmitting and receiving nodes. However, the ability to use time-of-flight measurements is dependent on the nodes having synchronized clocks. Thus in addition to estimating the positions of the nodes, the clock biases and bias rates must be estimated as well. A clock bias is an offset—for example, one clock might be running a second faster than another. The bias rate is how fast the clock bias is changing over time due to an imperfect oscillator. This work is motivated in part by an application where we envision multiple rovers exploring the lunar surface engaged in a collaborative science mission aided by a set of stationary beacons. The estimation algorithms must be of low enough computational complexity that they could be run on rover flight hardware. This motivates the Kalman Filter family of algorithms rather than other more computationally intensive approaches that have been proposed for robotic state estimation, such as batch non-linear least squares, maximum likelihood techniques, or recursive particle filters. Another driving requirement is that the rovers will be engaged in collaborative tasks which require that all rovers keep track of not only their own state estimates but state estimates of their fellow rovers. Lastly, it is assumed that communication packet sizes are limited and that rovers wish to share as much science data with each other as possible. GPS is not available on the Moon to provide either localization or timing information. Instead, a small

set of stationary beacons will be used as position references. Since there are no external references, both the clock biases and clock rates of the rovers and beacons will drift over time, which must be corrected to enable time-of-flight measurements. This work focuses on the development of Kalman Filter-based estimation techniques capable of estimating time-varying position and clock states in a distributed fashion.

The algorithms discussed here are designed to be run online, meaning that the nodes can use the algorithm to estimate their position and clock biases in real time. This type of problem is called Simultaneous Localization and Synchronization (SLAS) in the literature [14]. There is a large taxonomy of SLAS problems. Here we are interested in multiple-agent distributed SLAS with dynamic location parameters and dynamic clock parameters. Multiple-agent means that we are interested in estimating the location/clock parameters of all agents in the network as opposed to just one. Distributed implies that each node makes use of information received from neighbors and does computation locally rather than sending all of the measurements off to a single processor. Dynamic means that some of the agents are moving, so their velocities must be estimated as well; we call moving agents rovers and static agents beacons. Since the nodes have dynamic clock parameters we will be estimating time-varying clock biases and clock bias rates.

Yuan et al. [34] address distributed dynamic SLAS with time-varying clock offset, but their approach neglects clock bias rate. Etzlinger [13] shows the importance of estimating clock bias rate in addition to clock bias. The problem of estimating both time-varying clock biases and clock bias rates on dynamic nodes in a network is still relatively unexplored in the literature. A later work by Etzlinger [12] presents an algorithm for doing so, but says in their results section that they are not able to present a comparison with other methods since to the best of their knowledge there are no other SLAS methods for time-varying clock bias rate and clock offset. Since [12] was published there are few, if any, other comparable works. Their approach uses a hybrid particle-based Belief Propagation method with parametric representations of messages and beliefs. Their method is able to run in real-time on modern personal computers, but particle-filtering algorithms are typically too computationally intensive to run on flight computers which are not as capable as personal

computers in terms of processing power and memory. Additionally, Etzlinger's method tracks the full probability density function (pdf) of the state space using particles, which is problematic in a distributed estimation context because of the communication cost of sending particle sets. The requirements of low computational complexity and communication constraints motivate a Kalman Filter-based solution method.

This work uses one-way time-of-flight (TOF) as opposed to two-way ranging or time-of-arrival (TOA) localization. Two-way ranging capabilities are built in to many Commercial Off-The Shelf (COTS) ranging components like the DecaWave DW1000 [22] but such methods neglect frequency error, which introduces measurement error [11, 13], and require more frequent transmissions. TOA approaches are more favorable to two-step (synchronization then localization) approaches such as the application explored in [18].

The first distributed estimation method considered in this work is based on Distributed Data Fusion (DDF). DDF is a process in which agents collaboratively estimate quantities of interest by sharing information in the form of either state estimates or full state pdfs [5]. Uhlmann [32] introduced a technique called covariance intersection (CI) which conservatively fuses two state estimates; conservative means here that the estimated error covariance never underestimates the true estimated error covariance. In our case the nodes in the network regularly transmit joint estimates of their own and other agents' position and clock bias states to their neighbors, which are then fused using CI. One of the primary benefits of CI is that unknown cross-correlation terms need not be estimated [10]. Arambel [2] demonstrates the errors that arise if cross-covariances in shared state estimates are completely ignored and implements CI to mitigate the problem. Cattivelli and Sayed [7, 8] introduce diffusion strategies for distributed nonlinear estimation and go on to apply the method to wireless localization, but only consider position states and neglect clock states. Alanwar et al. [1] extend their work by allowing for distributed estimation of position, velocity, and clock states, but assume that the clock bias and bias rate are fixed. Both clock biases and clock bias rates drift over time [17], which motivates a CI-based algorithm for estimating position, velocity, and time-varying clock states.

The second method considered here is based on sharing measurements between agents using event-triggered (ET) estimation. ET is a technique used to reduce the amount of communication needed for estimation where only sufficiently surprising measurements are sent over the network. The network consists of remote sensor platforms and estimators that process data at different locations, where both sensors and estimators can be thought of as nodes through which neighbors exchange information. Unsurprising events provide implicit information since nodes know that the node constructing the measurement did not find the estimate surprising if they do not receive the measurement explicitly. This results in a set-valued measurement where a node does not know the exact value of a measurement, just that it was within a certain range. Shi et al. [29] shows that the truncated Gaussian produced by a set-valued measurement can be approximated by a Gaussian. In a later work, Shi [28] shows how these set-valued measurements can be incorporated into a Kalman Filter estimation framework. Some recent prior work has been done in the field of distributed event-triggered estimation [16], but no prior work has been done in applying event-triggering to position, navigation, and timing (PNT) estimation with pseudorange measurements. This thesis presents an algorithm for using measurement sharing methods to solve PNT estimation problems and discusses the effect of changing the event triggering threshold and clock noise parameters on localization accuracy.

The methodology in this paper builds on the work of Ouimet et al. [26] and Loefgren et al. [5], who presented a decentralized Kalman Filter-based algorithm that combines event-triggered measurement sharing with regular covariance intersection updates to resynchronize the network's state estimates. In particular, this thesis provides the foundation for extending the work in [26] and [5] to accomodate timing alongside position/navigation problems so that TOF ranging information for clock bias/error estimation can be integrated into both decentralized and event-triggered Kalman Filter state estimation algorithms.

The algorithm was tested in a simple simulation environment with three rovers and four beacons arranged to create a beneficial sensing geometry. The only measurements fed to the filter were pseudorange measurements with precision on the order of centimeters, since the clocks

were assumed to be low quality. The CI-based method nearly achieves the performance of the centralized filter used for comparison, but is under-confident in its estimates as expected of a conservative fusion method. The ET-based method offers reduced communication "for free" for small to moderate δ threshold values since there is little reduction in estimation accuracy. Both methods are quite sensitive to changes in the rate of communication, but are promising in that they are computationally cheap. Simulation results using a low-quality clock with measurements precise on the order of centimeters show that the CI method achieves a 2D position RMSE error of 0.5 m. The ET method significantly reduces required communication but has errors of about 1.8 m using a moderate triggering threshold of $\delta = 2$ m. The approaches developed here are promising but more investigations are needed into combining the two methods and relaxing the fully-connected-network assumption.

The remainder of the thesis is structured as follows. Chapter 2 covers the clock notation and model, pseudorange measurements, and the filter initialization method. Chapter 3 formally defines and motivates covariance intersection, walks through the steps of the estimation algorithm, and shows the results of a numerical simulation. Similarly, Chapter 4 introduces event triggering, shows the algorithm details, and discusses simulation results.

1.1 Thesis Contributions

The main research questions driving this work are:

Applying distributed estimation techniques There is a body of literature on distributed estimation leveraging covariance intersection and measurement sharing. How can existing distributed estimation techniques be applied to estimate position, velocity, and time-varying clock parameters to address aforementioned technical gaps for dynamic multi-platform applications?

Performance analysis How well are rovers able to localize themselves in simulations of the proposed lunar rover use case? Which clock/mission parameters have the largest effect

on positioning accuracy?

The major contributions of the thesis are:

Exploration of application of CI This thesis applies CI-based DDF to a sample PNT estimation use-case. We show algorithm implementation details, simulation results, and discuss the relative strengths and weaknesses of this method. Pseudorange measurements present a particular challenge since the measurement is a function of the state at two different times—we discuss a back-propagation method to handle this problem.

Exploration of application of ET This thesis additionally applies event-triggered measurement sharing to the same estimation problem and demonstrates the effect of varying clock/mission parameters on estimation accuracy in the sample lunar rover problem. We find that there is a large trade space for selecting the event-triggering parameter δ and that localization accuracy is very sensitive to the rate at which agents communicate.

Chapter 2

Background

In this chapter we will be introducing the estimation problem, measurement model, and filter initialization in support of the algorithm descriptions in subsequent chapters. First we will cover the formal problem statement to define the goal of the estimation algorithms. Pseudorange measurements are defined, and the importance of estimating clock biases is stressed by showing the magnitude of range bias introduced by clock dynamics over time.

2.1 Problem Definition

Let $p_i(k)$ be a quantity pertaining to agent i at time k. The position, velocity, and clock states of agent i's states advance according to

$$\vec{x}_i(k+1) = \vec{f}\,(\vec{x}_i(k), \vec{u}_i(k)) + \vec{w}_i(k) \tag{2.1}$$

with $\vec{w}_i(k) \sim \mathcal{N}(0, Q_i(k))$ and control input $\vec{u}_i(k)$.

Pseudorange measurements are a function of transmitter j's state and receiver i's state at two different times

$$\rho_{ij}(k) = h_{ij}\left(\vec{x}_i(k), \vec{x}_j(k')\right) + v_i(k) + v_j(k') \tag{2.2}$$

where $v_i(k) \sim \mathcal{N}(0, \sigma_{vi}^2(k))$, $v_j(k') \sim \mathcal{N}(0, \sigma_{vj}(k'))$, and $E[v_i(k)v_j(k')] = 0$ for $k \neq k'$. The pseudorange is a function of the state at two different times since the measurement depends on the position and clock states of the transmitter at time of transmission and on the position and clock states of the receiver at a later time of reception.

The network is assumed to be fully connected with no packet loss; i.e. if an agent transmits a message then it is guaranteed that all other agents will receive it. The agents are able to communicate timestamps in addition to state estimates and measurements. We do not consider message forwarding in this problem, meaning that i cannot transmit a message to j using l as an intermediary.

Stationary agents with known positions called beacons have two states: clock bias b and bias rate $\dot{b}$. So beacon B will have states

$$\begin{bmatrix} b_B \\ \dot{b}_B \end{bmatrix}. \tag{2.3}$$

Mobile agents called rovers contribute two-dimensional position (x, y) and velocity $(\dot{x}, \dot{y})$ in addition to clock states. For example, rover T will have states

$$\begin{bmatrix} x_T \\ y_T \\ \dot{x}_T \\ \dot{y}_T \\ b_T \\ \dot{b}_T \end{bmatrix}. \tag{2.4}$$

The full state vector $\vec{x}$ is a concatenation of all states in the network. Each agent uses a Kalman Filter (KF) to fuse local measurements with received messages/state estimates to construct a Minimum Mean Squared Error (MMSE) estimate of its own states and the states of all other agents.

Let $Y_i(1:k) = [\vec{y}(1)^T \ \cdots \ \vec{y}(k)^T]^T$ be a concatenation of all measurements constructed or received by agent i. Each agent seeks the MMSE estimate and associated Mean Squared Error covariance matrix

$$\hat{x}_i(k) = E\left[\vec{x}_i(k) | Y_i(1:k)\right] \tag{2.5}$$

$$P_i(k) = E\left[(\vec{x}_i(k) - \hat{x}_i(k))(\vec{x}_i(k) - \hat{x}_i(k))^T | Y_i(1:k)\right]. \tag{2.6}$$

These estimates are constructed locally by each agent using the measurements it has constructed as well as either state estimates received from other agents or measurements received from other agents.

2.2 Notation

Capital letters are used to denote agents or nodes in the network. By convention letters early in the alphabet (A, B, C, D) are used for stationary beacons while letters later in the alphabet (T, U, V) are used for rovers. Lower-case letters are used for the true time that agents take actions. For example, the time when agent d transmits a signal is written d^T, and the time that another agent receives that signal is written d^R.

The clocks are assumed to be imperfect and so the time of true event t as measured by agent I is written as $h_I[t]$. This measured time is equal to the true time of the event plus some clock offset at that time

$$h_I[t] = t + \Delta_I[t]. \tag{2.7}$$

To make the effect of the clock bias more understandable to humans (and to improve numerical stability) the bias is often written in terms of the bias b in meters by multiplying the clock offset by the speed of light, c. The measured time can then be expressed as

$$h_I[t] = t + \frac{b_I[t]}{c}. \tag{2.8}$$

We make the assumption throughout this work that agent A acts a clock reference, so all of the biases are implicitly relative to agent A's measured time. Similarly, when we consider the derivative of the measured time

$$\frac{\mathrm{d}h_I[t]}{\mathrm{d}t} = \dot{h}_I[t] = 1 + \frac{\dot{b}_I[t]}{c} \tag{2.9}$$

this is implicitly the rate of change relative to A's measured time.

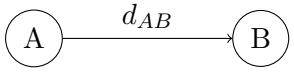

Figure 2.1: Time-of-flight measurement diagram

2.3 Pseudorange Measurements

Time-of-flight measurements are based on the fixed (given knowledge of the medium) speed of light. Agent A transmits a signal at time a^T which is received by B at time a^R. The distance between the two can be computed as the signal travel time $(a^R - a^T)$ times the speed of light

$$d_{AB} = c(a^R - a^T). \tag{2.10}$$

However, the two agents do not know the true time that these events occurred. The agents only have access to the measured time of these events, $h_A[a^T]$ and $h_B[a^R]$. Taking the difference of these two timestamps gives

$$\begin{aligned}
\rho_{BA} &= c\left(h_B[a^R] - h_A[a^T]\right) && (2.11)\\
&= c\left(\left(a^R + \Delta_B[a^R]\right) - \left(a^T + \Delta_A[a^T]\right)\right) && (2.12)\\
&= c\left(a^R - a^T\right) + c\Delta_B[a^R] - c\Delta_A[a^T] && (2.13)\\
&= d_{AB} + b_B[a^R] - b_A[a^T]. && (2.14)
\end{aligned}$$

The quantity ρ_{AB} is called the pseudorange from A to B. The pseudorange contains a measure of the distance between the two nodes biased by the clocks of the transmitting and receiving clocks. Even a small clock offset can have a large effect on the pseudorange measurement. Using $c \approx 3 \times 10^8\,\mathrm{m/s}$ we can see that a clock bias $\Delta = 1\,\mathrm{ns}$ produces a bias $b = c\Delta = 30\,\mathrm{cm}$. For applications that require sub-meter accuracy, clock biases must be estimated on the scale of single nanoseconds.

Equation 2.14 presents a slightly simplified story; when the agents are non-stationary then d_{AB} must be expanded. Let $\vec{r}_A[t]$ and $\vec{r}_B[t]$ be the vectors representing the positions of agents A and B at time t, and let $\|\cdot\|_2$ be the Euclidean norm. The pseudorange expression must then be

written as

$$\rho_{BA} = \left\|\vec{r}_B[a^R] - \vec{r}_A[a^T]\right\|_2 + b_B[a^R] - b_A[a^T]. \tag{2.15}$$

This expression shows that the pseudorange measurement depends on where the transmitter was at time of transmission and where the receiver was at time of receipt.

One of the challenges using pseudorange measurements is that the measurements generated from the same transmission are not independent. Consider the situation illustrated in Figure 2.2 where T transmits a signal received by A and B. T's message will include the timestamp $h_T[t^T]$ which allows the construction of measurements

$$\rho_{AT} = h_A[t^R] - h_T[t^T] = \left\|\vec{r}_A[t^R] - \vec{r}_T[t^T]\right\|_2 + b_A[t^R] - b_T[t^T] + cv_A[t^R] - cv_T[t^T], \tag{2.16}$$

$$\rho_{BT} = h_B[t^R] - h_T[t^T] = \left\|\vec{r}_B[t^R] - \vec{r}_T[t^T]\right\|_2 + b_B[t^R] - b_T[t^T] + cv_B[t^R] - cv_T[t^T]. \tag{2.17}$$

The last term of the two expression above shows that the two measurements have noise contributions from the same instantiation of the random variable $v_T[t^T]$, which means the measurements are correlated. The effects of this on each algorithm will be discussed in a later sections.

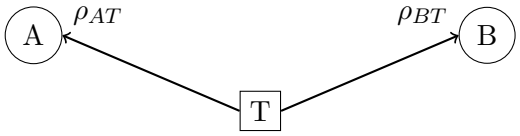

Figure 2.2: Transmission from agent T received by A and B

2.4 Clock Model

The clocks are modeled by a system of two linear differential equations with states $h_A[t]$ (the measured time) and $\dot{h}_A[t]$ (the derivative of the measured time with respect to a reference clock) collected in state vector

$$\vec{q}_A = \begin{bmatrix} h_A[t] \\ \dot{h}_A[t] \end{bmatrix}. \tag{2.18}$$

The clock dynamics are driven by Gaussian white noise added to the clock rate of change term

$$\frac{\mathrm{d}}{\mathrm{d}t}\vec{q}_A = \begin{bmatrix} \dot{h}_A[t] \\ w \end{bmatrix}. \tag{2.19}$$

The random variable is normally distributed $w \sim \mathcal{N}(0, \sigma_w^2)$. A value $\sigma_w = 51\,\mathrm{ns/s^2}$ was used for this work based on a value from [18]. Reference [18] also includes a higher-order term, noting that including a $\ddot{h}$ term is important while the clock is heating up; for this work we assume that the devices have been powered on for long enough to reach steady-state temperature. Reference [30] goes into more detail about the effect of temperature on the DecaWave ranging module which will be used as a reference device for our application. Additional information on clock modeling can be found in [15].

Figure 2.3 shows a sample trajectory of the clock bias over time using the selected noise parameter. After about 15 minutes, the bias is on the order of 60 km. This helps illustrate the importance of estimating clock bias when attempting to use time-of-flight measurements.

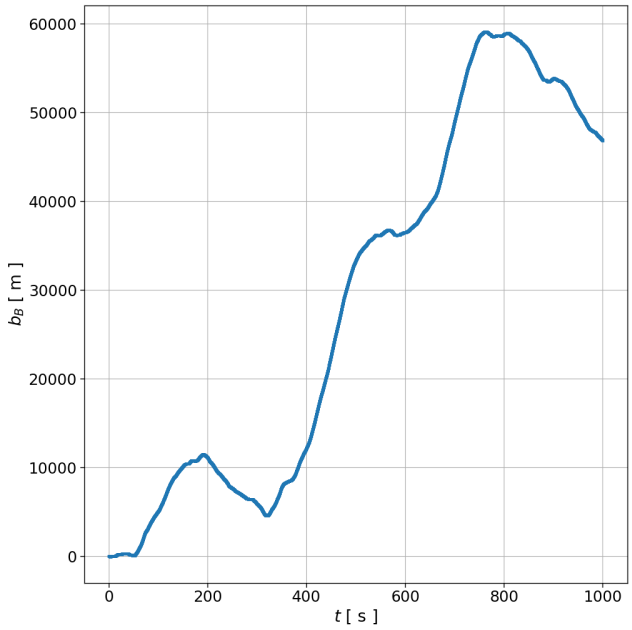

Figure 2.3: Sample clock bias drift over time

Each timestamp measurement includes some error as well due to finite timestamping precision

$$\vec{y} = \begin{bmatrix} 1 & 0 \end{bmatrix} \vec{q}_A + v \tag{2.20}$$

where $v \sim \mathcal{N}(0, \sigma_v^2)$. Using $\sigma_v = 0.13\,\text{ns}$ (from [18]) we can compute the uncertainty in each pseudorange measurement.

$$\rho = c(h_B[a^R] - h_A[a^T]) \tag{2.21}$$

$$\sigma_\rho^2 = c^2(2\sigma_v^2) \tag{2.22}$$

$$\sigma_\rho \approx 5.5\,\text{cm} \tag{2.23}$$

2.5 Filter Initialization

Consider a set of stationary beacons with known positions, along with a set of stationary rovers with unknown positions. We wish to estimate the position of the rovers, as well as the clock biases and bias rates of all of the agents, at the initial time, using pseudorange measurements. The measurement noise is assumed to be Gaussian, so a nonlinear least squares estimation approach is an appropriate choice for this problem.

Figure 2.4 shows a sketch of the geometry of the problem. Agent A is assumed to have a perfect clock, but the bias b and bias rate $\dot{b}$ of the other agents are to be estimated.

In this section square brackets $b[t]$ will be used to denote the quantity b at time t, while parentheses $b(t)$ are used as multiplication b times t. The state is to be estimated at some initial time t_0. The clock bias terms at a time t can be written as

$$b_V[t] = b_{V,0} + \dot{b}_{V,0}(t - t_0). \tag{2.24}$$

The pseudorange measurements then have the form

$$\rho_{UV} = \|\vec{r}_U - \vec{r}_V\|_2 + b_U[v^R] - b_V[v^T] \tag{2.25}$$

$$= \|\vec{r}_U - \vec{r}_V\|_2 + \left(b_{U,0} + \dot{b}_{U,0}(v^R - t_0)\right) - \left(b_{V,0} + \dot{b}_{V,0}(v^T - t_0)\right). \tag{2.26}$$

The true time difference $v^R - t_0$ is not known though; the agents are only able to use the measured time of those events. Computing the difference in times as measured by agent V gives

$$h_V[v^R] - h_V[t_0] = \left(v^R + \frac{b_{V,0}}{c} + \frac{\dot{b}_{V,0}}{c}(v^R - t_0)\right) - \left(t_0 + \frac{b_{V,0}}{c}\right) \tag{2.27}$$

$$= \left(1 + \frac{\dot{b}_{V,0}}{c}\right)(v^R - t_0). \tag{2.28}$$

The above makes the assumption that $\dot{b}$ is approximately constant over the considered time interval. Rearranging the above gives an expression for the true time difference which can be substituted in to the pseudorange equation

$$(v^R - t_0) = \frac{h_V[v^R] - h_V[t_0]}{1 + \frac{\dot{b}_{V,0}}{c}}. \tag{2.29}$$

Substituting in this expression and an analogous term for $(v^T - t_0)$ gives the pseudorange as

$$\rho_{UV} = \|\vec{r}_U - \vec{r}_V\|_2 + \left(b_{U,0} + \frac{\dot{b}_{U,0}}{1 + \frac{\dot{b}_{V,0}}{c}}\left(h_V[v^R] - h_V[t_0]\right)\right) - \left(b_{V,0} + \frac{\dot{b}_{V,0}}{1 + \frac{\dot{b}_{V,0}}{c}}\left(h_V[v^R] - h_V[t_0]\right)\right). \tag{2.30}$$

Given a measurement of ρ_{UV}, agent V can use the above expression to compute a predicted measurement $\hat{\rho}$ as a function of $\vec{x}_0$ and measured timestamps. We can then properly formulate the cost function. Given a set of M pseudorange measurements between different agents F and G, the cost function J is given by

$$J = \sum_{i=0}^{M} \left(\rho_{FG,i} - \hat{\rho}_{FG,i}(\vec{x}_0)\right)^2, \tag{2.31}$$

where $\rho_{FG,i}$ is the ith pseudorange measurement and $\hat{\rho}_{FG,i}$ is the prediction of the same pseudorange based on the state $\vec{x}_0$. The optimal estimate is an estimate which minimizes the cost function

$$\hat{x}_0 = \arg\min_{x_0} J. \tag{2.32}$$

There are many algorithms for solving this type of problem, such as the Gauss-Newton algorithm. The algorithm outputs an estimate of the initial state $\hat{x}_0$ and an estimate of the initial covariance P_0. There are many references discussing the full solution method; see [27], for example.

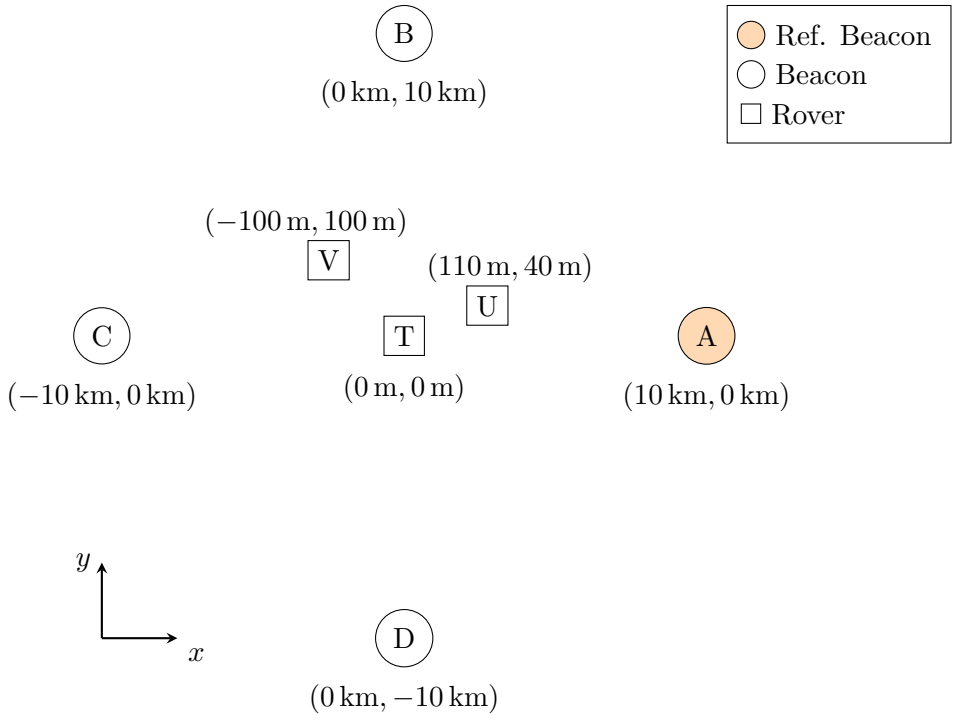

Figure 2.4: Sample geometry for nonlinear least squares problem setup

Chapter 3

Covariance Intersection Method

3.1 Covariance Intersection Theory

This chapter introduces one of the major thesis contributions: a PNT estimation algorithm leveraging covariance intersection (CI) to fuse information from other agents. First, a formal definition of CI and the motivation for conservative fusion are introduced. The algorithm is presented next, with emphasis on the necessary changes to a Kalman Filter to account for imperfect clocks, pseudorange measurements, and an additional fusion update. CI and related distributed estimation methods have been used in solving similar problems before; the novelty in this work is including time-varying clock parameters.

3.1.1 Covariance Intersection

An estimate $\hat{x}$ of quantity $\vec{x}$ with uncertainty covariance matrix P is called *consistent* if both of the following properties are satisfied [3].

$$E[\vec{x} - \hat{x}] = 0 \tag{3.1}$$

$$E[(\vec{x} - \hat{x})(\vec{x} - \hat{x})^T] = P \tag{3.2}$$

Where $E[\cdot]$ is the expected value operator. The actual Mean-Squared Error (MSE) of the left-hand side must match the filter covariance, P. An estimate is called *conservative* if the estimated uncertainty covariance matrix does not underestimate the true uncertainty covariance matrix [32]. A matrix P is said to be larger than a matrix Q if and only if $P - Q$ is positive semi-definite (PSD).

In this case, we will be fusing two covariance matrices P_A and P_B with unknown correlations. The true uncertainty is the estimation error that would be obtained if the unknown error correlations were taken into account.

Introduced by Uhlmann and Julier [32], covariance intersection is a method which, given two consistent estimates that may have unknown correlation between their estimation errors, provides a single consistent and conservative estimate by combining information from both. Crucially for this problem, covariance intersection makes no assumptions about the independence of the two estimates [32]. This feature is important for a scalable and practical solution to the distributed SLAS problem, since it allows agents to send information to each other in arbitrary communication topologies, and relaxes the otherwise computationally intractable requirement that all agents exactly track and remove dependent information that they previously shared with one another in order to avoid double-counting information.

First, we will consider fusing independent Gaussians. Given two independent estimates defined by their mean vectors $\hat{x}_A, \hat{x}_B$ and covariance matrices P_A, P_B, a single fused estimate can be computed as

$$P_C^{-1} = P_A^{-1} + P_B^{-1} \tag{3.3}$$

$$\hat{x}_C = P_C \left(P_A^{-1} \hat{x}_A + P_B^{-1} \hat{x}_B \right)^{-1}. \tag{3.4}$$

Covariance intersection, in contrast, includes a parameter $\omega \in [0, 1]$, which can be adjusted to scale the effect each component estimate has on the fused estimate

$$P_C^{-1} = \omega P_A^{-1} + (1 - \omega) P_B^{-1} \tag{3.5}$$

$$\hat{x}_C = P_C \left(\omega P_A^{-1} \hat{x}_A + (1 - \omega) P_B^{-1} \hat{x}_B \right)^{-1}. \tag{3.6}$$

The parameter ω is typically selected to minimize the trace or the determinant of the resulting covariance matrix. Figure 3.1 shows a comparison of the two different methods of data fusion. Given two estimates (blue and orange), naively assuming independence gives the small red covariance ellipse, while using covariance intersection gives the larger green covariance ellipse.

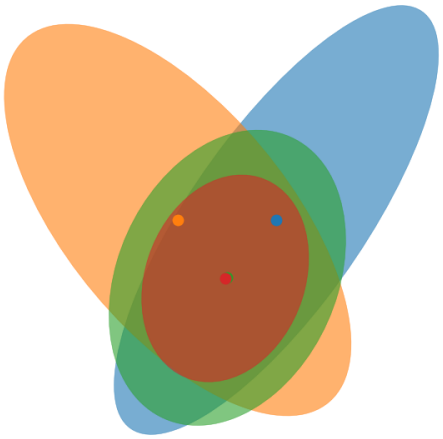

Figure 3.1: Fusion of two estimates (blue and orange) assuming independence (red) or using covariance intersection (green)

In this method, agents will send their state estimates to each other. A simple example will motivate the use of covariance intersection. Consider Figure 3.2. Assume that the agents have not communicated before this moment, so all sets of information are independent.

(1) A transmits its state estimate $(\hat{x}_A, P_A)$, which is received by T. T fuses the independent estimates together $(\hat{x}_T, P_T)$; T's covariance estimate is now $P'_T = (P_A^{-1} + P_T^{-1})^{-1}$

(2) Similarly, B receives the estimate from A; B's covariance estimate is now $P'_B = (P_A^{-1} + P_B^{-1})^{-1}$

(3) Now B transmits to T. T's options are either to fuse the B's assuming independence or using covariance intersection

   (a) Assuming independence, T's fused covariance would be

$$P''_T = (P'_T + P'_B)^{-1} \tag{3.7}$$

$$= (P_A^{-1} + P_T^{-1} + P_A^{-1} + P_B^{-1})^{-1} \tag{3.8}$$

$$= (\underbrace{2P_A^{-1}}_{\text{double counting!}} + P_T^{-1} + P_B^{-1})^{-1} \tag{3.9}$$

This is problematic because the estimated covariance erroneously double-counts the information from A.

(b) Using covariance intersection, the fusion looks like

$$P_T'' = (\omega P_T' + (1-\omega)P_B')^{-1} \tag{3.10}$$

$$= (\omega P_A^{-1} + \omega P_T^{-1} + (1-\omega)P_A^{-1} + (1-\omega)P_B^{-1})^{-1} \tag{3.11}$$

$$= (P_A^{-1} + \omega P_T^{-1} + (1-\omega)P_B^{-1})^{-1} \tag{3.12}$$

This shows that covariance intersection is able to avoid the problem of double-counting information when the two estimates to be fused are not independent.

The above is a very simple illustration of an effect called rumor propagation, which becomes a problem when agents do not know whether information from an agent is novel, or just parroted from a different agent. Covariance intersection is used to solve the rumor propagation problem in this work. Reference [5] covers rumor propagation more extensively.

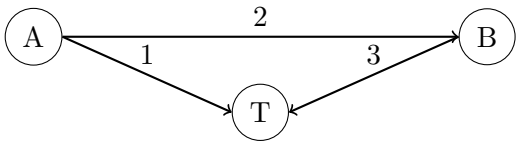

Figure 3.2: Simple communication diagram to illustrate rumor propagation. B and T receive a message from A, then T receives a message from B

Another advantage of covariance intersection is that it does not depend on all agents using a Kalman Filter to obtain state estimates. If one of the agents is running a batch algorithm or Particle Filter, it can still output a mean estimate and error covariance matrix, allowing that information to be fused with the network. Similarly, it is easy to update the capabilities of individual agents without needing to update the software on all of the rest of the agents. If one rover is equipped with a lidar then its state estimates will become more accurate and that improved state awareness will propagate to the rest of the network even though no other agents know how to handle lidar measurements.

3.2 Covariance-Intersection SLAS Algorithm

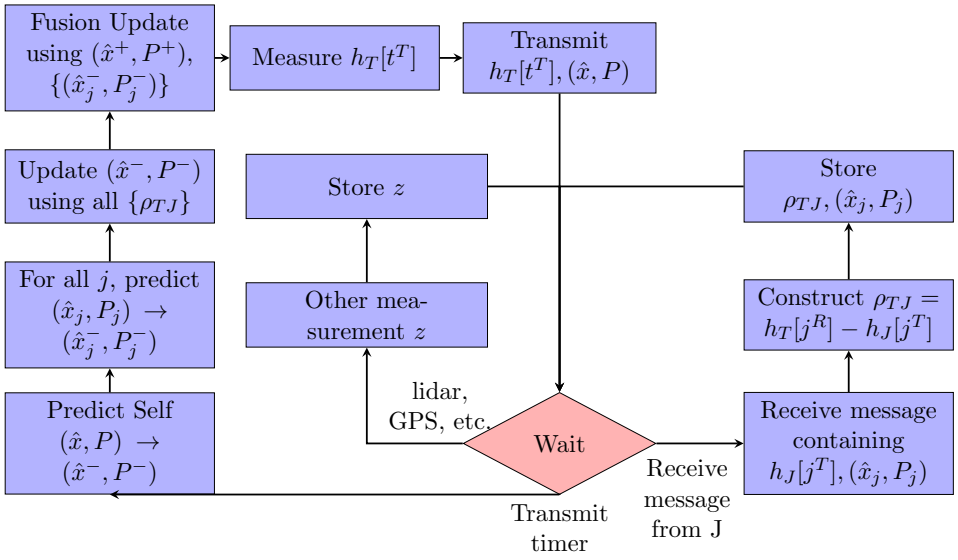

Figure 3.3: Covariance intersection SLAS Algorithm Diagram

Figure 3.3 shows a high-level overview of the algorithm running on each agent. The outer left loop shows the regular transmissions. Based on a timer, the agent will predict its state estimate forward to the present time, propagate all of the received estimates to the present time, run a measurement update, and then do covariance intersection to fuse all of the estimates together. The agent then measures the current time and includes that timestamp in the transmission along with its fused state estimate. While waiting to transmit, the agent will frequently be interrupted either by receiving a transmission from another node (the right loop) or by another type of measurement (inner left loop). The following subsections will go through the details of each of the loops.

3.2.1 Receiving Messages

First we will look at the right loop of the algorithm. When a message from generic agent J is received, the message will contain the timestamp when J transmitted the message ($h_J[j^T]$) as well as J's estimate at transmit time $(\hat{x}_j, P_j)$. An additional timestamp will be created when the

message is received ($h_T[j^R]$) which allows for the construction of a pseudorange measurement

$$\rho_{TJ} = h_T[j^R] - h_J[j^T]. \tag{3.13}$$

This pseudorange measurement and the received state estimate are added to a queue for processing by the filter algorithm.

3.2.2 Filter Details

Filter Propagation For purpose of discussion, only the states x and $\dot{x}$ will be considered, but the same method applies to the $y/\dot{y}$ states and $b/\dot{b}$ states. The system of differential equations describing the motion is linear and can be written

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix} x \\ \dot{x} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \dot{x} \end{bmatrix}. \tag{3.14}$$

We can discretize the above equation to define the transition from x_k to x_{k+1}. The equation is separable, and with the assumption that $\dot{x}$ is constant over the time interval considered we can derive the transition of state x in terms of the change in time Δt as

$$\frac{\mathrm{d}x}{\mathrm{d}t} = \dot{x} \tag{3.15}$$

$$\int \mathrm{d}x = \int \dot{x}\mathrm{d}t \tag{3.16}$$

$$x_{k+1} - x_k = \dot{x}(t_{k+1} - t_k) \tag{3.17}$$

$$x_{k+1} = x_k + \dot{x}\Delta t, \tag{3.18}$$

and can similarly derive the transition for $\dot{x}$

$$\frac{\mathrm{d}\dot{x}}{\mathrm{d}t} = 0 \tag{3.19}$$

$$\dot{x}_{k+1} - \dot{x}_k = 0 \tag{3.20}$$

$$\dot{x}_{k+1} = \dot{x}_k. \tag{3.21}$$

The problem is that the clocks onboard each agent are imperfect, so Δt is unknown. The differential equation can be reformulated in terms of the measured time $h[t]$ as

$$\frac{\mathrm{d}x}{\mathrm{d}h[t]}\frac{\mathrm{d}h[t]}{\mathrm{d}t} = \dot{x} \tag{3.22}$$

$$\int \mathrm{d}x = \int \dot{x}\frac{\mathrm{d}t}{\mathrm{d}h[t]}\mathrm{d}h[t]. \tag{3.23}$$

Taking the time derivative of the definition of the measured time gives the necessary extra term

$$h[t] \equiv t + \frac{b}{c} \tag{3.24}$$

$$\frac{\mathrm{d}h[t]}{\mathrm{d}t} = 1 + \frac{\dot{b}}{c}. \tag{3.25}$$

So the propagation in terms of the difference in measured time $\Delta h[t]$ can be written as

$$x_{k+1} = x_k + \frac{\dot{x}_k}{1 + \frac{\dot{b}_k}{c}}\Delta h[t] \tag{3.26}$$

where $\Delta h[t] = h[t_{k+1}] - h[t_k]$. For many applications $\dot{b} \ll c$ so $\frac{\dot{x}_k}{1+\frac{\dot{b}_k}{c}} \approx \dot{x}_k$. The truth-model simulation uses the nonlinear prediction Equation 3.26, while the filter uses the linear equation

$$x_{k+1} = x_k + \dot{x}_k \Delta h[t]. \tag{3.27}$$

during the prediction step.

For a given agent I, the filter is run immediately before transmission. $\Delta h_I[t]$ can be computed by subtracting the previous measured transmission time from the current transmission time

$$\Delta h_I[t] = h_I[i_k^T] - h_I[i_{k-1}^T]. \tag{3.28}$$

Predicting the received estimates forward is slightly more tricky. The state estimate was created at time $h_J[j^T]$, it was received at time $h_T[j^R]$, and is being processed at $h_T[t^T]$. The time difference between receiving and the current time is the amount of time that the state estimate was waiting in the queue to be processed: $\tau_{\text{wait}} = h_T[t^T] - h_T[j^R]$. The other duration accounts for the time that the state estimate was traveling between agent J and agent T. This can be

approximated by computing the estimated distance between the two and dividing by the speed of light $\tau_{\text{dist}} = \frac{\hat{d}_{TJ}}{c}$. This is an approximation since the propagation time actually depends on the difference between the transmitter at transmit time and the receiver at receive time rather than the distance at any particular instant. In this application the agents are moving fairly slowly so the agent motion during the transmission time can be neglected. For satellite applications the distance term becomes much more important and iterative methods may be necessary to account for how the agents move while the signal is in transmission. GPS texts such as [23] cover the topic in detail.

Measurement Update After the propagation step we have the filter mean and covariance at the present time $(\hat{x}_i^-(k), P_i^-(k))$; we will fuse the set of received pseudorange measurements $\{\rho_{IJ}\}$ and other measurements $\{z\}$. The measurements are nonlinear, which requires a nonlinear modification to the Kalman Filter. For this work we use the Unscented Kalman Filter (UKF) [19] measurement update. The idea behind a UKF is generating a set of representative points from the state estimate (called "sigma points"), running those points through the nonlinear measurement function, and approximating the posterior statistics using those points.

Algorithm 1 walks through the steps of the measurement update. In steps 1–4 the sigma points are generated using a Cholesky decomposition (to find the matrix square root) and run through the measurement function $\tilde{h}$ which is covered in more detail in Algorithm 2. In lines 5–6 all of the pseudorange measurements and any additional measurements (e.g. GPS or lidar) are concatenated together. In lines 7–8 the measurements are predicted using a weighted sum of transformed sigma points. We use $p^{[i]}$ to denote the ith element of a list. See [19] for weights w_m and w_c. The uncertainty in the predicted measurement and the cross-covariance matrix between the states and measurements are found in lines $9-12$. Lastly, the Kalman Gain is computed and used to estimate the state estimate mean and covariance in lines 15–18.

The pseudorange prediction depends on the states at two different time instants

$$\hat{\rho}_{IJ} = \left\|\hat{r}_I[j^R] - \hat{r}_I[j^T]\right\|_2 + \hat{b}_I[j^R] - \hat{b}_J[j^T]. \tag{3.29}$$

Algorithm 1: UKF measurement update within CI-based algorithm

input : State estimate $\hat{x}_i^-(k), P_i^-(k)$, set of pseudorange measurements $\{\rho_{IJ}\}$, $\{z\}$
output: Updated state estimate $\hat{x}_i^-(k), P_i^-(k)$

1 Generate sigma points;
2 $\chi(k) = \begin{bmatrix}\hat{x}_i^-(k) & \hat{x}_i^-(k) + \gamma\sqrt{P_i^-(k)} & \hat{x}_i^-(k) - \gamma\sqrt{P_i^-(k)}\end{bmatrix}$;
3 Transform sigma points using measurement function;
4 $\Upsilon(k) = \tilde{h}(\chi(k))$;
5 Concatenate pseudorange measurements and other measurements;
6 $\vec{y}(k) = \begin{bmatrix}\rho_{IA} & \rho_{IB} & \cdots & z_{\text{GPS}} & z_{\text{lidar}}\end{bmatrix}^T$;
7 Predict measurement;
8 $\hat{y}(k) = \sum_{i=0}^{2n} w_m^{[i]} \Upsilon^{[i]}(k)$;
9 Predicted measurement uncertainty;
10 $P_{yy}(k) = \sum_{i=0}^{2n} w_c^{[i]} \left(\Upsilon^{[i]}(k) - \hat{y}(k)\right)\left(\Upsilon^{[i]}(k) - \hat{y}(k)\right)^T + R(k)$;
11 Find cross-covariance between state and measurement;
12 $P_{xy}(k) = \sum_{i=0}^{2n} w_c^{[i]} \left(\chi^{[i]}(k) - \hat{x}_i^-(k)\right)\left(\Upsilon^{[i]}(k) - \hat{y}(k)\right)^T + R(k)$;
13 Compute filter gain;
14 $K(k) = P_{xy}(k) P_{yy}^{-1}$;
15 Update state estimate;
16 $\hat{x}_i^+(k) = \hat{x}_i^-(k) + K(k)\left(\vec{y}(k) - \hat{y}(k)\right)$;
17 Update state estimate uncertainty;
18 $P_i^+(k) = P_i^-(k) - K(k) P_{yy} K^T(k)$;

After the propagation step we have the state estimate at time i^T. Since the Kalman Filter measurement update depends on the sensitivity of the measurement to the state at the present time, the predicted measurement has to be a function of only current states. The solution is to back-propagate the present time estimate $\hat{x}[i^T]$: by τ_{wait} to get $\hat{x}[j^R]$, and by $\tau_{\text{wait}} + \tau_{\text{dist}}$ to get $\hat{x}[j^T]$. These steps are shown in lines 3–11 of Algorithm 2. Each sigma point is a full state vector. We denote a quantity extracted from the qth sigma point with a $[q]$ superscript. For example, $\hat{r}_I^{[q]}$ is the estimate of the position of I, $\vec{r}_I$, from the qth sigma point. The outer loop iterates over all of the sigma points $\chi^{[q]}$ and builds the predicted measurement vector $\Upsilon^{[q]}$ based on that state.

Fusion Update Recall that each received message contains the transmitting agent's estimate of the states at transmit time. At this point we have the updated local state estimate $(\hat{x}^+, P^+)$ as well as a set of received state estimates propagated to the present time $\{(\hat{x}_j^-, P_j^-)\}_j$. We wish to fuse these together using covariance intersection. A slight modification must be made

Algorithm 2: Measurement function to produce predicted measurements from set of sigma points

input : Set of sigma points χ
output: Set of predicted measurements Υ

foreach q **from** 0 **to** $2n$ **do**
  foreach agent J **from set of pseudorange measurements do**
    Compute back-propagation times;
    $\tau_{\text{wait}} = h_I[i^T] - h_I[j^R]$;
    $\tau_{\text{dist}} = \frac{\left\|\hat{r}_I^{[q]}[i^T] - \hat{r}_J^{[q]}[j^T]\right\|_2}{c}$;
    Propagate receiver states back;
    $\hat{r}_I[j^R] = \hat{r}_I^{[q]}[i^T] - \hat{\dot{r}}_I^{[q]}[i^T]\tau_{\text{wait}}$;
    $\hat{b}_I[j^R] = \hat{b}_I^{[q]}[i^T] - \hat{\dot{b}}_I^{[q]}[i^T]\tau_{\text{wait}}$;
    Propagate transmitter states back;
    $\hat{r}_J[j^T] = \hat{r}_J^{[q]}[i^T] - \hat{\dot{r}}_J^{[q]}[i^T]\,(\tau_{\text{wait}} + \tau_{\text{dist}})$;
    $\hat{b}_J[j^T] = \hat{b}_J^{[q]}[i^T] - \hat{\dot{b}}_J^{[q]}[i^T]\,(\tau_{\text{wait}} + \tau_{\text{dist}})$;
    Compute estimated pseudorange measurement;
    $\hat{\rho}_{IJ} = \left\|\hat{r}_I[j^R] - \hat{r}_J[j^T]\right\|_2 + \hat{b}_I[j^R] - \hat{b}_J[j^T]$;
    Append $\hat{\rho}_{IJ}$ to $\Upsilon^{[q]}$;
  end
  foreach additional measurement z **do**
    Compute estimated measurement;
    $\hat{z} = h(\chi^{[q]})$;
    Append $\hat{z}$ to $\Upsilon^{[q]}$;
  end
end

to the CI equation to allow for the fusion of more than two estimates. The fused covariance and state vector are given as

$$P_{\text{fused}}^{-1} = \sum_i \omega_i P_i^{-1} \tag{3.30}$$

$$\hat{x}_{\text{fused}} = P_{\text{fused}} \left(\sum_i \omega_i P_i^{-1} \hat{x}_i \right). \tag{3.31}$$

Instead of the selection of a single scalar ω value, a set of $\{\omega_i\}$ values must be selected to minimize the trace or determinant of P_{fused} with the constraint that $\sum_i \omega_i = 1$. There are many methods for solving this type of constrained optimization problem; here we used an off the shelf "Trust Constrained" algorithm based on [9].

Consider a simplified scenario with only two agents, A and B. Additionally assume that the

bias in clock A is zero ($b_A = 0$) so that the only unknowns are B's clock bias b_B and the distance between the two d_{AB}. With these assumptions, the pair of pseudorange measurements ρ_{AB} and ρ_{BA} reduce to

$$\rho_{AB} = d_{AB} - b_B \tag{3.32}$$

$$\rho_{BA} = d_{AB} + b_B. \tag{3.33}$$

Figure 3.4 shows how the pair of measurements can be used to construct an estimate in this scenario: the estimate is the intersection of the lines, and the estimate uncertainty is determined by the measurement uncertainty.

For this method we do not have both measurements though. All that we have is one half of the pseudorange pair and the state estimate of the other agent. Figure 3.5 presents a sketch of the information that B has to work with. B will repeatedly get measurements of ρ_{BA}; state errors parallel to ρ_{BA} are unobservable, (this will be discussesd more in the event-triggering section) and so B's uncertainty in the estimate will be extended along that direction. The received estimate from A is uncertain in the other direction. Fusing the two estimates allows the algorithm to obtain the correct state estimate since the estimate from A supplies the other half of the information.

3.2.3 Additional Measurement

Lastly, we return to the inner left loop of the diagram to talk about additional measurements. For example, one rover might have access to lidar or GPS-like measurements. These measurements will be processed in the Kalman Filter measurement update step and so the transmitted state estimate $(\hat{x}, P)$ will be more accurate. One benefit of using covariance intersection to share information between agents is that the other agents do not need to know how to handle the additional measurement type. The measurement information has already improved the state estimate and the extra information will propagate through the network. Rovers having better position awareness improves the accuracy of clock estimates as well, so all agents in the network get better location and clock estimates just by adding a sensor to one rover.

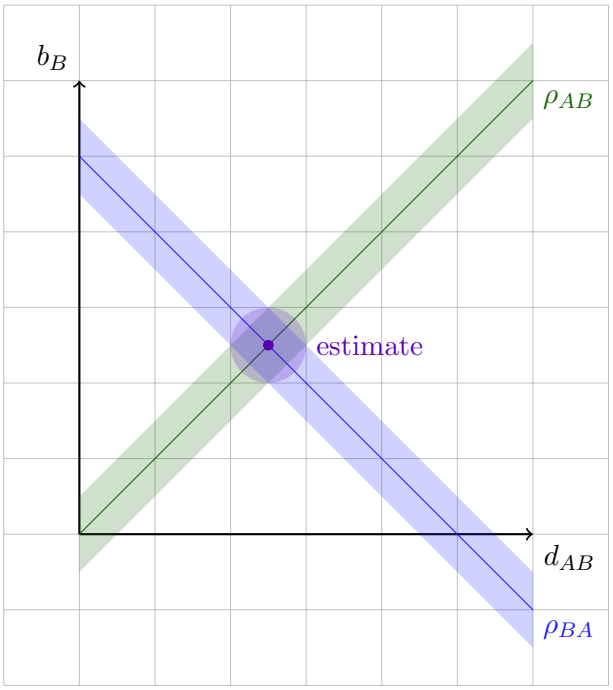

Figure 3.4: Diagram showing the intersection of ρ_{BA} and ρ_{AB}

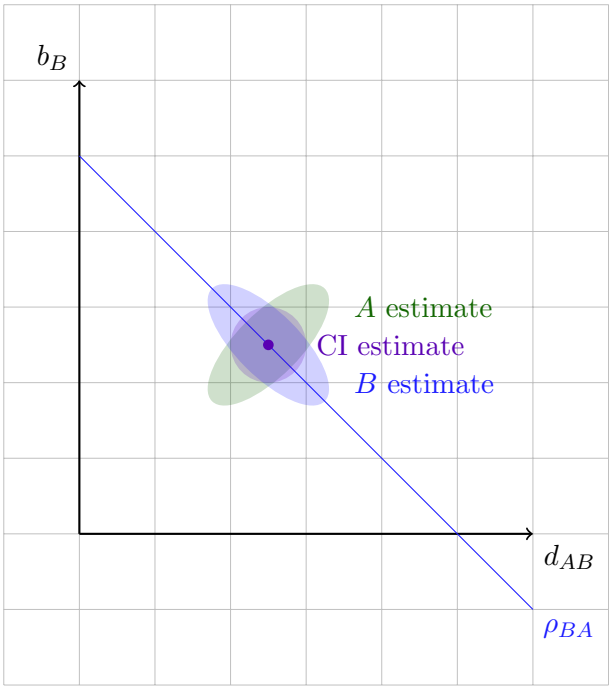

Figure 3.5: Diagram of covariance intersection to find distance-bias estimate

3.3 Numerical Simulation

3.3.1 Simulation Setup

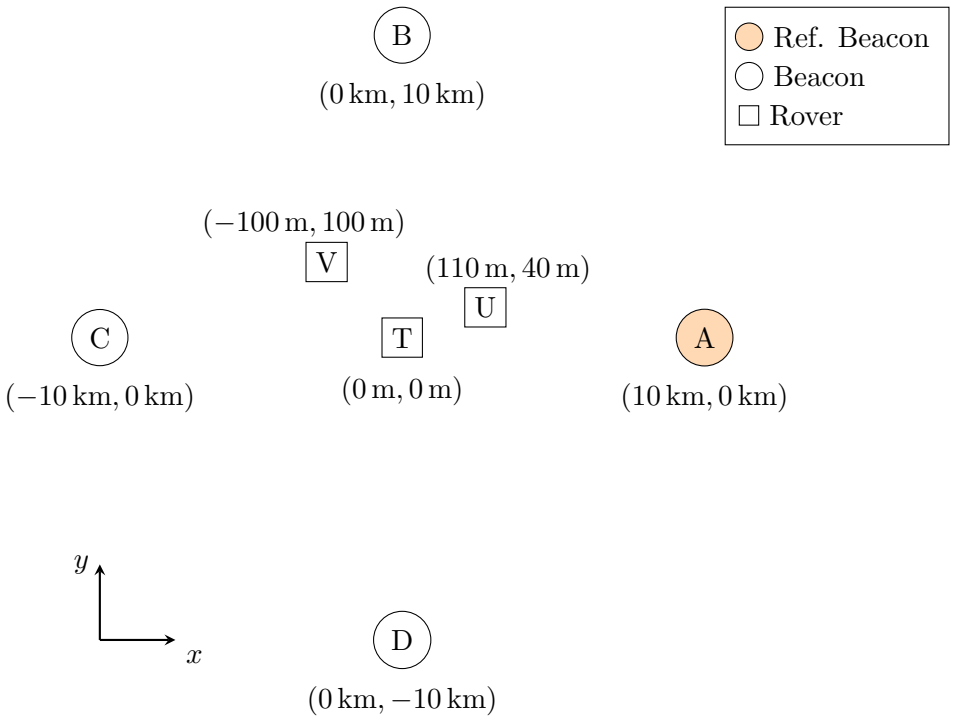

Figure 3.6: Numerical simulation diagram showing sample mobile rovers (squares) as well as stationary beacons (circles). One of the agents (orange circle) acts as a temporal reference.

Figure 3.6 shows a sample diagram of the experimental setup. Four beacons are set up at known locations several kilometers in each direction with one agent (Agent A by convention) acts as the temporal reference, meaning that the bias and bias rate of its clock are fixed to be zero. Starting at random locations, the rovers drive around in simple patterns near the center. The rovers' control laws (i.e. the control inputs as a function of time and estimated state) are known to all agents. The network is assumed to be fully connected, meaning that transmissions from any agent are received by all.

All agents share the same state vector, which contains all of the states to be estimated in the whole network. All of the non-temporal-reference agents contribute bias b and bias rate $\dot{b}$ states. Additionally, each of the rovers contributes two-dimensional position (x, y) and velocity

$(\dot{x}, \dot{y})$ states. All of these states have the relevant agent's name as a subscript (e.g. Agent B's clock bias states are b_B and $\dot{b}_B$.)

Agents transmit in their assigned transmission window. Each agent has one transmission window per cycle; the window has a duration of 0.1 s in this simulation. The transmission order is alphabetical, so Agent A will transmit, then 0.1 s later Agent B will transmit, followed by Agent C 0.1 s later. This is known as a Time-Division Multiple Access (TDMA) schedule, designed to prevent signal interference introduced by agents transmitting at the same time.

While the algorithm is able to support additional types of measurement such as GPS or lidar measurements, only agent–agent pseudorange measurements are used for this simulation.

3.3.2 Results

Figure 3.7 shows the clock bias estimation results and reveals an effect of the TDMA schedule on estimation accuracy. Agent C has just done a measurement update with a batch of measurements involving its own clock bias so it has a good estimate of its own bias states. Similarly, C most recently received a message from B with its state estimate and so has a good estimate of B's clock bias as well. However, a full communication cycle has passed since C has heard from D and so B's estimate of D's clock states is a lot more uncertain.

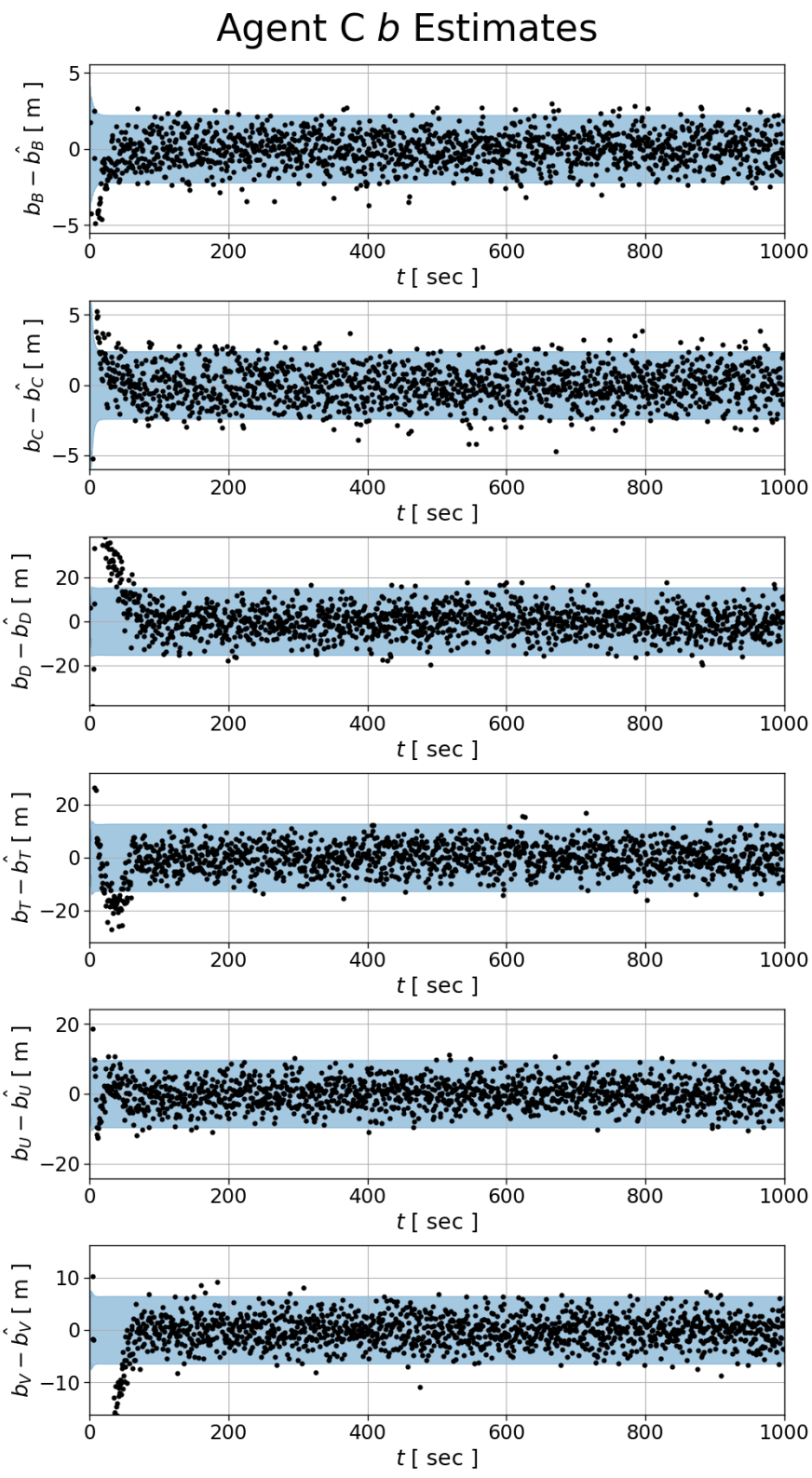

Figure 3.7: Agent C clock bias errors, using CI algorithm

Rover U's estimate of all rover position and velocity states is shown in Figure 3.8. All agents have estimates of all states in the network—and so similar position and velocity plots exist for all other agents—but these results are representative. In a Monte Carlo truth-model simulation, the average steady-state 2D position RMSE for all agents is about 0.2 m on average. The reported RMSE value only considers errors after steady state has been reached.

Figure 3.8 also shows the uncertainty obtained by the centralized filter. The fictional centralized filter instantly receives all measurements as soon as they are created and acts as a comparison case. The difference between the centralized filter and the decentralized filter is essentially the cost of approaching the problem in a distributed fashion. Note that in the position estimation error results the errors are closely bounded by the centralized uncertainty, meaning that the filter achieves approximately the accuracy of the centralized filter but is under-confident. Under-confidence is consistent with expectations since CI fuses information conservatively.

3.3.3 Discussion

The filter is able to do surprisingly well in terms of position and velocity estimation. Estimation of position using pseudorange measurements depends on clock bias estimation accuracy. The RMSE error for each of the biases is about 4.0 m, while the RMSE values for positions are less than a meter. In this problem the stochastic changes in clock states are significantly greater than the stochastic changes in position/velocity states, which explains the difference in estimation accuracy.

An additional advantage of using CI for data fusion is that the problem of pseudorange measurement non-independence is easily addressed. In the example shown in 3.9, the measurements

$$\rho_{AT} = h_A[t^R] - h_T[t^T] = \left\|\vec{r}_A[t^R] - \vec{r}_T[t^T]\right\|_2 + b_A[t^R] - b_T[t^T] + cv_A[t^R] - cv_T[t^T], \tag{3.34}$$

$$\rho_{BT} = h_B[t^R] - h_T[t^T] = \left\|\vec{r}_B[t^R] - \vec{r}_T[t^T]\right\|_2 + b_B[t^R] - b_T[t^T] + cv_B[t^R] - cv_T[t^T] \tag{3.35}$$

are correlated with each other since they share the same instantiation of $v_T[t^T]$ from the transmission timestamp. Without CI, this would present a problem when A and B share state estimates with each other—the correlated information would have to be accounted for. However, CI guarantees

that the fused estimate will be consistent even in the presence of unknown correlations between the estimates so no additional work is needed.

One of the large sources of uncertainty is in assuming constant velocity when propagating received states forward and in the measurement prediction equation (propagating by τ_{wait}). The shortcoming of the constant-velocity assumption must be compensated for by adding process noise. This limitation will be discussed further at the end of the next section in the discussion of varying the communication frequency. Despite this additional process noise, the algorithm works significantly better when we wait to have a batch of pseudorange measurements and state estimates rather than running the filter whenever a message is received. This is likely because a smaller covariance ellipse is obtainable by running covariance intersection on a batch of estimates rather than by sequentially fusing estimates. For example, in general, fusing $(\hat{x}_A, P_A)$ with $(\hat{x}_B, P_B)$ and fusing the result with $(\hat{x}_C, P_C)$ will not produce the same result as fusing all three at the same time. This is because in the latter case the optimization algorithm is able to vary the weighting of all three pieces of information simultaneously to find the minimum determinant, whereas in the prior case the ratio of $(\hat{x}_A, P_A)$ to $(\hat{x}_B, P_B)$ is fixed. This is in contrast to a linear Kalman Filter where sequential updates should give the same result.

Changing clock noise parameters σ_w and σ_v as well as the frequency of communication have a large effect on estimation results. These effects will be explored at the end of the next section since changing these parameters affects the CI algorithm and the ET algorithm in the same way.

One of the major disadvantages of covariance intersection is the cost in terms of communication. The full length n state vector and unique elements of the covariance matrix must be transmitted in every message. The following chapter explores one method for reducing the amount of communication necessary for the same problem statement.

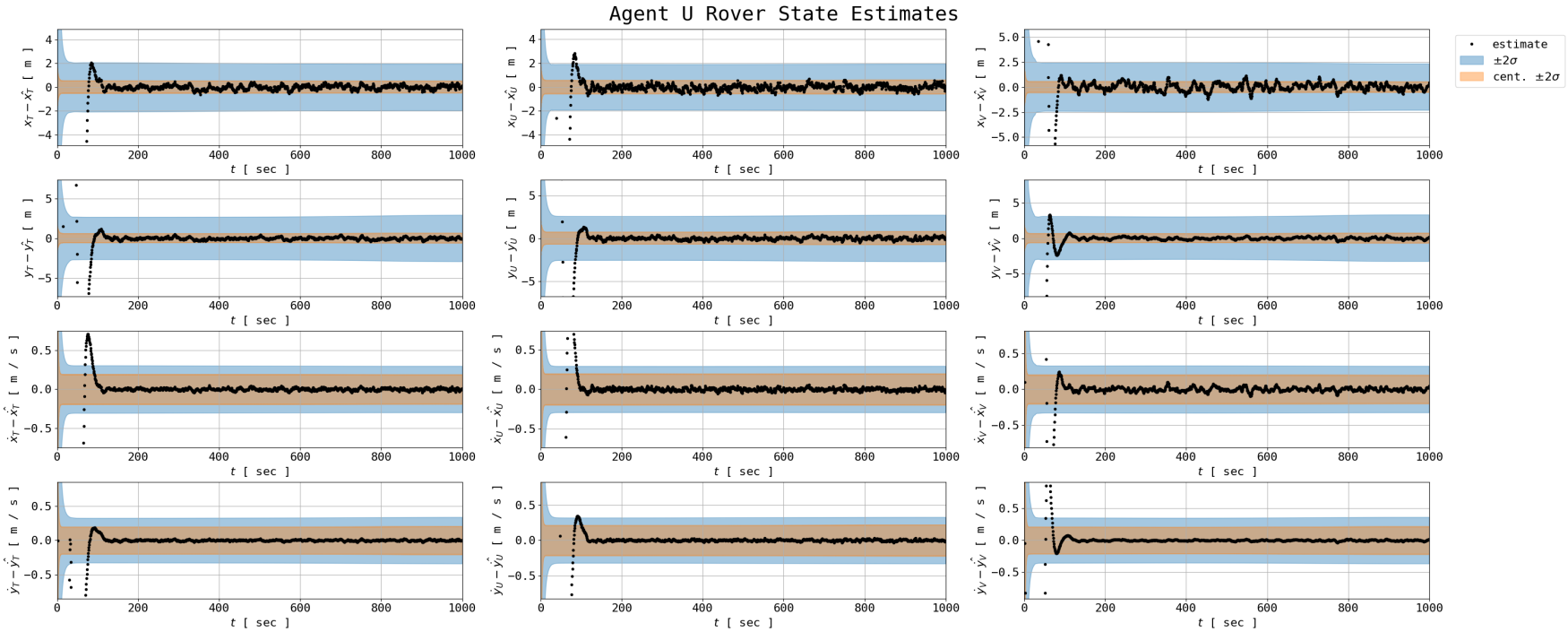

Figure 3.8: Agent U rover state errors, using CI algorithm (CI estimates and 2σ bounds shown in black dots with blue shading; centralized filter 2σ shown in gold)

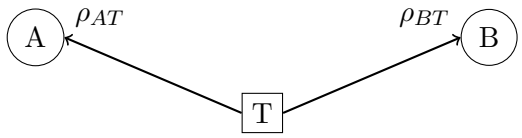

Figure 3.9: Transmission from agent T received by A and B

Chapter 4

Event Triggering Method

This chapter describes the second set of novel contributions of this work. In particular, it shows the use of event-triggered (ET) measurement sharing methods applied to PNT estimation with pseudorange measurements. The first section motivates the use of ET methods and introduces the idea. The ET-based algorithm is introduced with emphasis on the Kalman Filter measurement update modifications necessary to accommodate set-valued measurements. The algorithm is validated in a numerical simulation with a discussion of communication savings, network observability, and the effect of varying clock noise parameters.

This is a fundamentally different approach to decentralized cooperative PNT than the previous chapter. We are no longer sending state estimates at all but sending measurement data between filters. This has the benefit of avoiding unnecessary information loss due to being overly conservative (as happens with CI), but limits the amount of information that can be spread through the network. The spread of information is limited only to neighbors, since received measurements are not re-transmitted. Loefgren and Ahmed [21] combined CI and measurement sharing to obtain the benefits of both, but did not incorporate imperfect clocks or timing estimation. While the previous chapter considered extending CI to the clock estimation domain, this chapter looks at how measurement sharing can incorporate timing information.

4.1 Event-Triggered Estimation Theory

The covariance intersection method from the previous section works well, but requires large packets of information to be sent with each transmission. For a length n state vector, each transmission will contain: 1 floating point value for the timestamp, n floating point values for the mean vector, and $\frac{n(n+1)}{2}$ floating point values for the unique elements of the covariance matrix. The simulation closing the previous section had $n = 24$ states, which means that each transmission contained 325 floating point values. In many applications (e.g. [21]) communication payload size is limited. Ideally we would like to be able to reduce the amount of information contained in each packet without sacrificing too much estimation accuracy. This would allow for the newly freed space to be used for transmitting science data between agents.

One method of reducing communication is sending measurements only when they are sufficiently surprising. When an agent receives a measurement and finds that the residual is larger than some predefined δ bound then the agent will explicitly re-transmit the measurement out to other agents (Fig. 4.1). If the measurement is not surprising according to this metric then the measurement will not be sent. Since other agents expecting the measurement know the δ bounds, they are able to fuse the information that the measurement was within those bounds. This means that even an absence of information is information, allowing for fewer measurements to be sent.

4.2 Event-Triggering SLAS Algorithm

We make the same set of assumptions here about problem setup and communication topology. Consider a group of N robot agents, some mobile and some at known fixed positions, that are connected in a fixed communication and sensing topology. As before, each agent runs its own local Kalman Filter and follows a fixed transmission schedule. For the purpose of this paper it is assumed that the network is fully connected—all agents talk to all other agents in the network. This assumption is not strictly necessary, but it allows for all states to remain observable at each node. This assumption can be relaxed in future work when measurement sharing and CI are combined

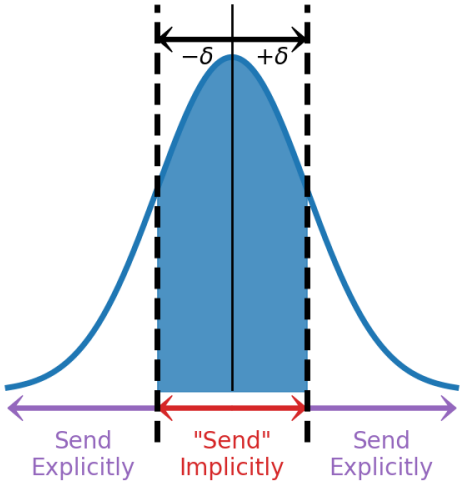

Figure 4.1: Event-triggering δ bounds

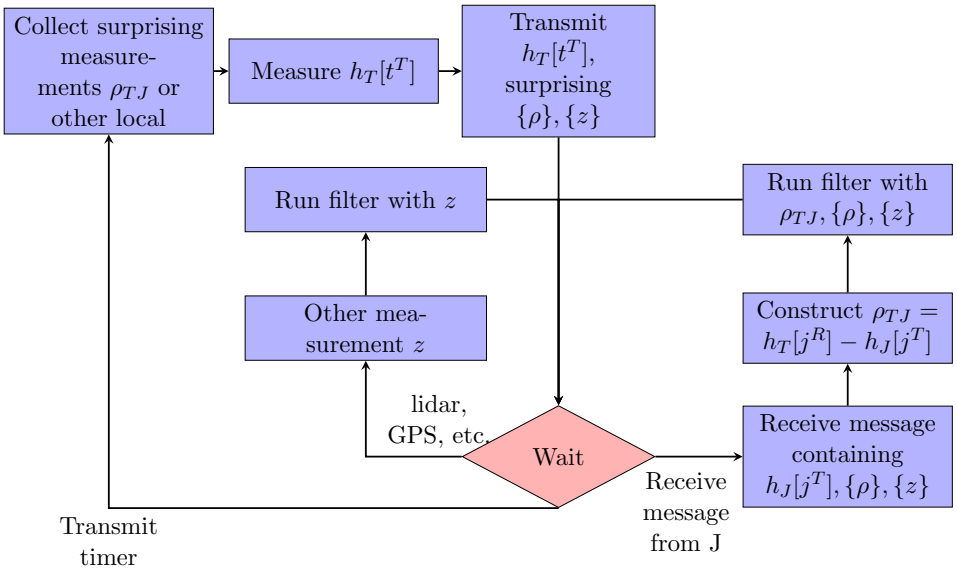

Figure 4.2: Event triggered SLAS Algorithm Diagram

together as in the work by Ouimet, et al. [26] and Loefgren et al. [21].

The algorithm diagram for the event-triggering SLAS method is shown in Figure 4.2. One major difference to call out is that state estimates are no longer transmitted; instead only surprising measurements are transmitted to other agents (outer left loop). When a message is received (right loop) the message contains the transmit timestamp and the set of pseudorange/other measurements that the transmitting agent found surprising. A timestamp is constructed, then the filter is run using

the local pseudorange measurement and the sets of surprising measurements. This method is still able to support non-pseudorange measurements (inner right loop), but now all neighboring nodes need to know how to handle all types of measurements since the measurements are transmitted out to the network. The filter runs the outer left transmit loop regularly according to a timer, but the filter will be run several times during each waiting interval whenever a measurement is received.

The ET algorithm has fewer steps than the CI algorithm; the prediction step is the same, but there are no received states to predict and no fusion update step. The measurement update now has to be able to process both explicit and implicit measurements, which is what we will discuss next.

4.2.1 Measurement Update with Explicit/Implicit Measurements

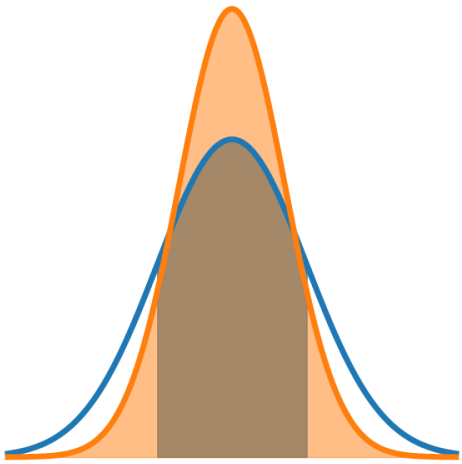

Figure 4.3: Fitting a Gaussian (orange) to a truncated Gaussian (non-truncated region highlighted in blue)

There are three types of measurement that the local filter for each agent has to know how to handle: local measurements, explicitly communicated measurements, and implicitly communicated measurements.

Local measurements are those created onboard, (ρ_{TJ} or z in the diagram). These measurements are processed as discussed in the previous chapter. The measurement residual on local

measurements determines whether it will be transmitted explicitly or implicitly. If the measurement residual is larger than some δ bound (sufficiently surprising) then it will be sent implicitly, otherwise it will be "sent" implicitly.

Explicit measurements are received from other agents. The handling of explicit measurements is similar to local measurements—we still have an actual value, so the typical Kalman Filter update equation works. The only difference is that explicit measurements are not checked to see whether they are surprising, and are not re-transmitted. (e.g. if agent I receives an explicit measurement from agent J, then agent I does not re-transmit this message to some other agent M.) Some implementations of event-triggered estimation involve re-transmission of received measurements, but that is not considered here.

Agents always receive messages from each other—we are assuming no dropped packets here—but in the case of an implicit measurement the actual value is missing. The receiving agent knows that the measurement exists, but does not know the value of the measurement. It is only known that the measurement was within the predefined δ bounds. If there is a possibility of dropped packets then there is some ambiguity about whether a measurement was sent implicitly or dropped. The issue is not addressed here, but see [20] for the necessary algorithm modifications to handle the possibility of dropped packets. In our case, this means that agents know whether or not they have received sent data packets, and that data packets are encoded so as to indicate the presence/absence of specific explicit measurements, along with the values of explicit measurements which are present. Loefgren [21] considers how coding schemes can be constructed using both uncompressed double-precision representations for messages to transmit explicit measurements with implicit data flags as well as compressed integer representations using piecewise quadratic quantization for restricted message sizes. We consider only uncompressed messaging here.

Since only the bounds of the measurement are known, the measurement distribution is then a Gaussian with truncated support (the blue region in Figure 4.3). Properly fusing a Gaussian prior state pdf and a truncated Gaussian measurement likelihood function via Bayes' rule would lead to a non-Gaussian posterior pdf which cannot be completely recovered in closed form via the

Kalman Filter. Shi [29] shows that fitting a Gaussian to a truncated Gaussian is a reasonable approximation. Given a truncated Gaussian $f(x) \sim \mathcal{N}(\mu, \sigma^2)$ with support for $x \in [\mu - \delta, \mu + \delta]$, we can approximate the distribution using a Gaussian $g(x) \sim \mathcal{N}(\mu + \bar{z}, (1 - \vartheta)\sigma^2)$. Computing $\bar{z}$ and ϑ will be discussed later in this section.

Measurements with a point value (i.e. local or explicit measurements) are fused using the normal measurement update equations

$$\hat{x}^+ = \hat{x}^- + Kr \tag{4.1}$$

$$P^+ = (I - KH)P^-, \tag{4.2}$$

where K is the Kalman Gain, r is the measurement residual, and H is the measurement sensitivity matrix.

Fusing implicit measurements uses a modified form of the update equations

$$\hat{x}^+ = \hat{x}^- + K\bar{z} \tag{4.3}$$

$$P^+ = (I - \vartheta KH)P^-. \tag{4.4}$$

The value $\vartheta \in [0, 1]$ describes the shape of the fitted Gaussian and directly affects the the updated covariance matrix P^+. In the limiting case of $\vartheta = 1$, the fitted Gaussian collapses to a point and the implicit measurement update equation is identical to the normal measurement update equation. In the other limiting case of $\vartheta = 0$ the fitted distribution is the same as the measurement distribution. This means that the δ bound is too large and no information can be obtained from the implicit measurement. In this case the implicit update collapses to $P^+ = P^-$. Said differently, setting $\delta = 0$ yields $\vartheta = 1$ while in the limit as $\delta \to \infty$ then $\vartheta \to 0$.

The next question to address is how each agent is able to know what implicit information to fuse. For example, when A receives no measurement from B, then all that A knows is that B did not find the measurement surprising. To be able to do anything with that information, A needs to have knowledge of what B's state estimate is. For this application we make the assumption that the network is fully connected, and that all agents will therefore have approximately the same state

estimate. This is a reasonable assumption because, in a fully-connected network, all agents will have access to all measurements in the network either by constructing them locally or receiving them implicitly/explicitly from neighbors. To continue the example, B will send a message implicitly if

$$|\rho_{BC} - \hat{\rho}^B_{BC}| \leq \delta, \tag{4.5}$$

where $\hat{\rho}^B_{BC}$ is Agent B's prediction of the measurement. When Agent A receives no measurement, it uses the approximation $\hat{\rho}^A_{BC} \approx \hat{\rho}^B_{BC}$ and is able to bound the measurement as

$$-\delta + \hat{\rho}^A_{BC} \leq \rho_{BC} \leq \delta + \hat{\rho}^A_{BC}. \tag{4.6}$$

We are now ready to discuss the computation of $\bar{z}$ and ϑ. Define the following:

$$Q_e = HP^- H^T + R, \tag{4.7}$$

$$\nu^- = -\frac{\delta}{\sqrt{Q_e}}, \tag{4.8}$$

$$\nu^+ = \frac{\delta}{\sqrt{Q_e}}, \tag{4.9}$$

where R is the measurement noise matrix. For this method the measurements are processed one at a time, so the quantity Q_e will be a scalar and the $\sqrt{Q_e}$ is defined as normal. Let $\phi(\cdot)$ be the Gaussian probability density function and $Q(\cdot)$ be the Gaussian cumulative distribution function.

$\bar{z}$ expresses the difference between the mean of the truncated Gaussian and the mean of the fitted Gaussian. There is no difference in this case since, by assumption, the predicted measurement of the receiving agent is the same as the predicted measurement of the transmitting agent. Therefore $\bar{z}$ will always be zero in this case. The fitted Gaussian variance adjustment can be found as

$$\vartheta = \frac{\nu^+\phi(\nu^+) - \nu^-\phi(\nu^-)}{Q(\nu^+) - Q(\nu^-)}. \tag{4.10}$$

For applications that do not make the fully-connected assumption, the computation of $\bar{z}$ and ϑ is more complicated; see [21] and [26].

There are many other triggers in ET estimation beyond using measurement residuals. Another method is by triggering based on whether a new measurement is outside of some δ bound of

the previous explicit measurement. This triggering strategy avoids making the $\hat{\rho}^A_{BC} \approx \hat{\rho}^B_{BC}$ approximation since there is no longer any ambiguity about the center of the $\pm\delta$ bounds. This alternate trigger is not used here but could be explored in future work.

4.3 Numerical Simulation

The same simulation from the previous section is used here as well. The main figure of merit defining success for this algorithm is the localization accuracy. Figure 4.4 shows the average 2D position RMSE values of all rovers in the network for different δ values from a 30-run Monte Carlo simulation. Table 4.1 shows the corresponding average error value and percent of measurements sent implicitly. Lower δ values perform better as expected since the filter has more information to work with. One of the main takeaways is that there is little reduction in performance for small or moderate δ values. At a high δ value of 10 m the filter experiences a more noticeable drop in performance but only sends about 20% of measurements explicitly.

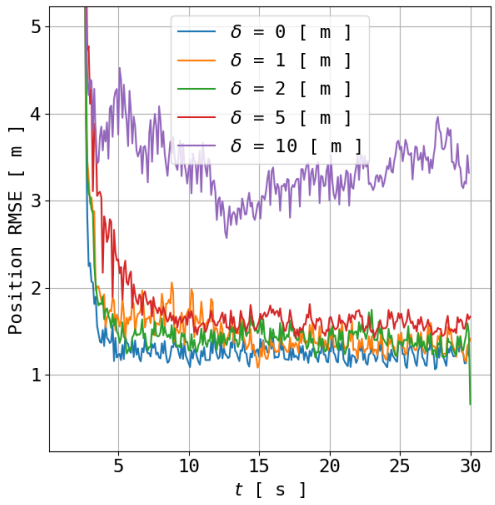

Figure 4.4: Average 2D position RMSE values for different δ values over time

We will again examine agent U's estimates of all rover position and velocity states, but the results for the other agents are similar. Figure 4.5 shows the position and velocity estimation errors along with the $\pm 2\sigma$ estimated uncertainty and the $\pm 2\sigma$ bounds for the centralized filter as a comparison. The comparison of uncertainty bounds between the decentralized and centralized filter

Table 4.1: Communication savings and average 2D position for different δ values

δ [m]	Average 2D position RMSE [m]	% Implicit Measurements
0	1.6	0
1	1.7	10
2	1.8	21
5	2.0	50
10	3.4	81

allows us to see the effect of fusing implicit measurements—the state uncertainty grows, accounting for the spiky blue bounds in the plot. The centralized filter has access to all measurements explicitly and so its uncertainty stays fairly constant.

4.3.1 Communication Savings

Each measurement sent implicitly represents a payload communication savings of the size of a floating point number on the computers used. For this work 4 bytes per value is assumed. Table 4.2 shows the number of implicit and explicit measurements transmitted by each agent in a typical simulation. Each implicitly-transmitted measurement contributes to the communication savings, so agents that send more measurements implicitly have larger savings. While the majority of measurements are still sent explicitly, each agent is able to reduce the amount of information sent by at least 30 kB over the 15 min simulation time. Agent A sends fewer explicit messages than any of the other agents. This makes sense since A acts as a temporal reference; A has no clock states, so it is less likely for it to be surprised by a local measurement since the pseudorange measurements are only affected by one clock's drift rather than two.

One of the main benefits of event-triggering is that the mission designer can set different δ values at different times during the mission. While the rovers are hundreds of meters apart and tasked with collecting science data and sharing it between themselves then it makes sense to set δ very high so rovers only have a rough idea of where they are. If rovers are engaged in a collaborative task such as exploring a small crater together then setting δ much lower allows the rovers to maintain accurate estimates of each other and avoid crashing into each other.

Table 4.2: Number of implicit/explicit messages sent and communication savings for event-triggering with $\delta = 2\,\text{m}$ during a 15-minute simulation

Agent	# Implicit	# Explicit	Communication Savings [kB]
A	16608	34674	66
B	10380	40932	41
C	10302	41010	41
D	10128	41178	40
T	10488	40818	41
U	9792	41508	39
V	9378	41916	37

4.3.2 Pseudorange Measurement Non-Independence

We again consider the problem of pseudorange measurement non-independence. In the case of a centralized filter, having correlated measurements could easily be handled by accounting for the correlation in the measurement noise matrix R. Decentralized estimation makes the problem much more challenging given the time delays involved. For example, suppose that A and B both receive transmissions from T and construct ρ_{AT} and ρ_{BT}. One option would be for A to wait to incorporate ρ_{AT} until it has received ρ_{BT} from B some time later. However, as we have seen previously, the uncertainty in the measurement increases as it becomes more out of date given how quickly the clock biases change. The alternative is to incorporate measurements as soon as they are received. Nonlinear filters are approximate methods already and so the measurement correlation can be neglected at the loss of some information. In practice, the benefit obtained from waiting for the measurement and accounting for the measurement correlation does not outweigh the increased accuracy from incorporating the measurement as soon as possible. Further research is needed in this area.

4.3.3 Observability

One of the major differences between the CI method and the ET method is the observability of the network as a whole to any given agent. Consider the simple scenario shown in Figure 4.6. Agent T is able to communicate with agent U and U is able to talk to agent V, but T and V cannot

communicate directly. As discussed previously, both halves of a pair of pseudorange measurements are needed (e.g. ρ_{TU} and ρ_{UT}) in order to determine the two unknowns (distance and relative clock bias). In the ET method, agent T constructs ρ_{TU} locally and receives ρ_{UT} and ρ_{UV} explicitly or implicitly from agent U. However, since messages are not forwarded, the information contained in ρ_{VU} will never reach agent T. This means that the network as a whole will always be unobservable to T since there is an ambiguity between V's position and clock bias. When using the CI method on the other hand, there is no problem. ρ_{VU} will be constructed by V and fused into its state estimate; the state estimate will then pass through U and reach T.

The network diameter is the maximum number of steps to get between any given pair of nodes. When using a CI-based method in a non-fully-connected network the diameter has a large effect on the estimation accuracy. The clock bias drift is significant (on the order of tens of meters per second) and so mission designers must consider the effect that additional communication hops will have on clock bias estimation accuracy.

When using measurement-sharing methods, the position and clock states of an agent are unobservable to agents not directly connected to it. Thus, for effective measurement sharing, the network needs to be fully connected, or nodes need regular assistance from other agents in the form of CI updates.

4.3.4 Variation of Clock Noise Parameters

There are two clock noise parameters: perturbing noise $w \sim \mathcal{N}(0, \sigma_w^2)$ which affects the dynamics and error from finite timestamping precision $v \sim \mathcal{N}(0, \sigma_v^2)$ which distorts each timestamp measurement.

Figure 4.7 shows the effect of varying the noise perturbing the clock dynamics, σ_w. Recall that $\sigma_w = 51\,\mathrm{ns/s^2}$ (the red line) was used in earlier simulations. The lower bound is, of course, setting $\sigma_w = 0$. This means that the bias b and bias rate $\dot{b}$ will maintain their initial values. The only position error results from random noise added to the control input. The value of $\sigma_w = 51\,\mathrm{ns/s^2}$ is already fairly high, so most clocks will produce results between the red line and blue line shown

in the figure.

Figure 4.8 shows that varying the timestamp precision σ_v has surprisingly little effect on localization results. The yellow line for $\sigma_v = 0.13\,\text{ns}$ is the noise parameter used in the other simulations. Decreasing the precision by an order of magnitude significantly reduces localization capabilities during the first few seconds of the simulation, but there is no difference in the steady state estimation accuracy. This is due to the large number of measurements relative to the motion of the rover. The stochastic perturbations of the rovers' motions are small, and so the filters are able to keep track of their locations even with less accurate measurements. If the rovers' motions were more highly perturbed or measurements arrived at a lower rate then the timestamping precision would have a larger effect.

4.3.5 Variation of Transmission Schedule

All agents transmit in alphabetical order at times separated by Δt. (i.e. agent A transmits at time $t = 0$, agent B transmits at time $t = \Delta t$, agent C at time $t = 2\Delta t$, etc.) The localization accuracy is highly dependent on the parameter Δt, since this corresponds directly to how much information enters the network. Figure 4.9 shows how the default value of $\Delta t = 0.1\,\text{s}$ used in simulations above compares to higher and lower rates of communication. Cutting the wait in half produces a moderate increase in localization capabilities, while increasing the delay even to $\Delta t = 0.3\,\text{s}$ increases the RMSE to nearly 10 m.

In summary, the δ parameter should be selected to balance estimation needs with desired communication savings. The clock parameter σ_w has a moderate impact on estimation accuracy, but purchasing a higher quality clock can ameliorate the effect. The transmission window Δt significantly affects the localization capabilities and should be carefully chosen. The timestamping precision σ_v is of less importance for the scenarios considered in this problem.

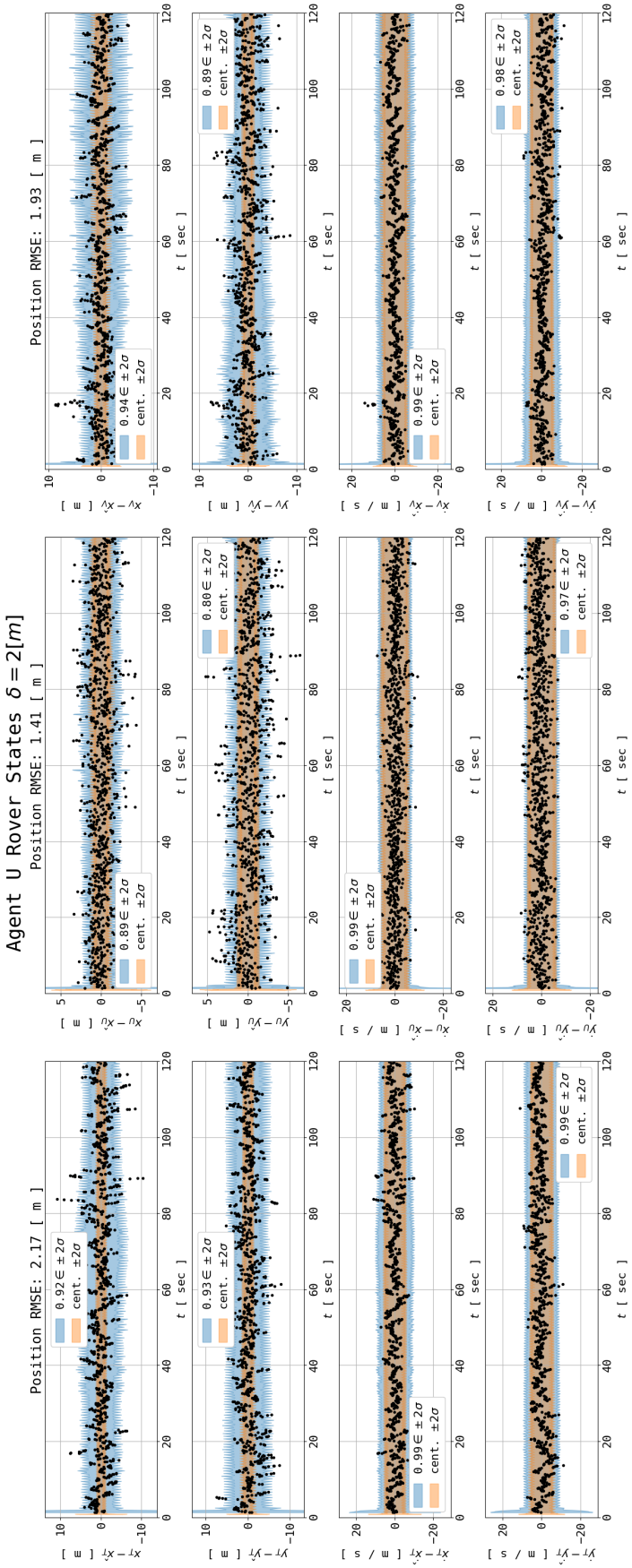

Figure 4.5: Agent U rover state errors, using ET algorithm with $\delta = 2\,\mathrm{m}$

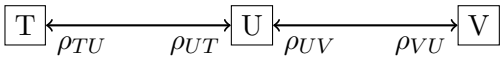

Figure 4.6: Simple communication diagram to illustrate observability differences between CI and ET methods

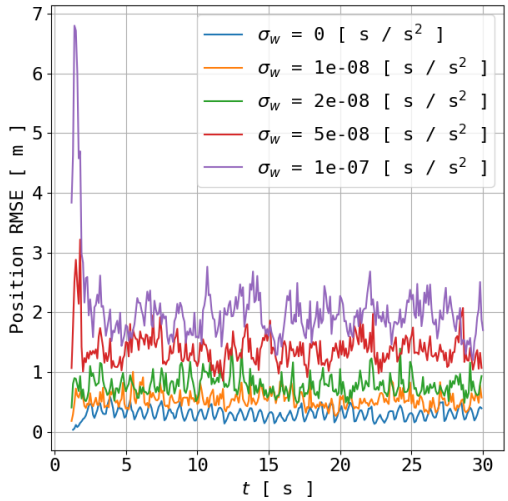

Figure 4.7: Average 2D position RMSE values for different σ_w values over time

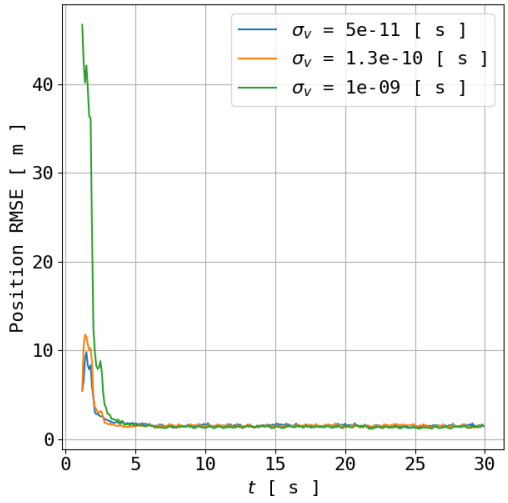

Figure 4.8: Average 2D position RMSE values for different σ_v values over time

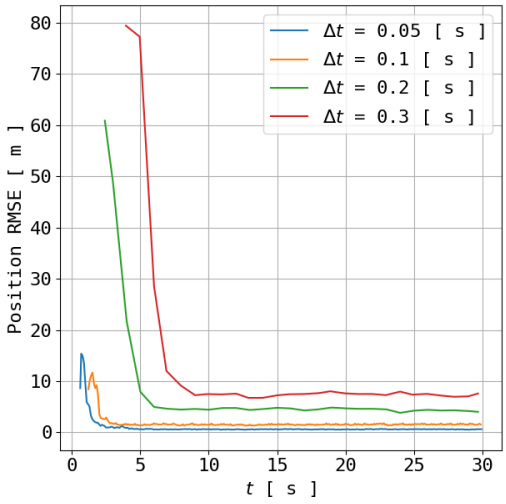

Figure 4.9: Average 2D position RMSE values for different transmission window Δt values over time

Chapter 5

Conclusion

This work introduces two solution methods for solving dynamic distributed localization and synchronization problems. The first method is based on sharing state estimates using covariance intersection and is useful in situations which require each platform to maintain full network state observability (i.e. knowledge of their own and all other platforms' position and clock states) with sparse communication topologies. The second method is based on event-triggered measurement sharing and is useful for reducing communications overhead as well as avoiding information losses from conservative state estimate fusion. These methods are currently limited by assumptions about the communication topology, but are applicable to use-cases where the network is fully connected. The covariance intersection method is generally more accurate with a 2D position RMSE of 0.5 m but requires sending large packets of information across the network. The event-triggering method allows for a significant reduction in communication by only sending surprising measurements and has a 2D position RMSE of 1.8 m for a sample triggering threshold of $\delta = 2\,\text{m}$. Simulation results show that low-to-moderate δ values allow for significant communication savings while incurring only a small loss in estimation accuracy. Additionally, results are shown to be sensitive to transmission rate and the clock drift parameter, but relatively insensitive to the timestamping precision.

One of the main limitations of this work is that only the fully-connected network case is explored. Other works employing a similar solution method, namely [21], drop the fully-connected assumption. In that work only a few agents have access to inertial position information through GPS. The information propagates through the network and all rovers are able to obtain inertial

estimates even if they are one or two steps removed from the inertial reference. This work is a proof-of-concept to show that distributed estimation techniques are applicable to distributed position, navigation, and timing problems. Future work will include a combined CI/ET method which leverages the benefits of both, as in [21] and [26].

This work is partially motivated by application to a NASA STTR considering a team of lunar rovers supported by a set of stationary beacons. The methods introduced here are currently being implemented in a higher-fidelity simulation environment and in hardware to demonstrate the application of these algorithms to future exploration missions. Additionally, future work will include comparisons to other algorithms such as [12].

Bibliography

[1] Amr Alanwar, Henrique Ferraz, Kevin Hsieh, Rohit Thazhath, Paul Martin, João Hespanha, and Mani Srivastava. D-slats: Distributed simultaneous localization and time synchronization. In Proceedings of the 18th ACM International Symposium on Mobile Ad Hoc Networking and Computing, Mobihoc '17, New York, NY, USA, 2017. Association for Computing Machinery.

[2] P. O. Arambel, C. Rago, and R. K. Mehra. Covariance intersection algorithm for distributed spacecraft state estimation. In Proceedings of the 2001 American Control Conference. (Cat. No.01CH37148), volume 6, pages 4398–4403 vol.6, 2001.

[3] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. Estimation with applications to tracking and navigation: theory algorithms and software. John Wiley & Sons, 2004.

[4] R. M. Buehrer, H. Wymeersch, and R. M. Vaghefi. Collaborative sensor network localization: Algorithms and practical issues. Proceedings of the IEEE, 106(6):1089–1114, 2018.

[5] Mark E Campbell and Nisar R Ahmed. Distributed data fusion: Neighbors, rumors, and the art of collective knowledge. IEEE Control Systems Magazine, 2016.

[6] J. Cano, S. Chidami, and J. L. Ny. A kalman filter-based algorithm for simultaneous time synchronization and localization in uwb networks. In 2019 International Conference on Robotics and Automation (ICRA), pages 1431–1437, 2019.

[7] F. S. Cattivelli and A. H. Sayed. Diffusion strategies for distributed kalman filtering and smoothing. IEEE Transactions on Automatic Control, 55(9):2069–2084, 2010.

[8] F. S. Cattivelli and A. H. Sayed. Distributed nonlinear kalman filtering with applications to wireless localization. In 2010 IEEE International Conference on Acoustics, Speech and Signal Processing, pages 3522–3525, 2010.

[9] Andrew R Conn, Nicholas IM Gould, and Philippe L Toint. Trust region methods. SIAM, 2000.

[10] Zili Deng, Peng Zhang, Wenjuan Qi, Jinfang Liu, and Yuan Gao. Sequential covariance intersection fusion kalman filter. Information Sciences, 189:293–309, 2012.

[11] Adwait Dongare, Patrick Lazik, Niranjini Rajagopal, and Anthony Rowe. Pulsar: A wireless propagation-aware clock synchronization platform. In 2017 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), pages 283–292. IEEE, 2017.

[12] Bernhard Etzlinger, Florian Meyer, Franz Hlawatsch, Andreas Springer, and Henk Wymeersch. Cooperative simultaneous localization and synchronization in mobile agent networks. IEEE Transactions on Signal Processing, 65(14):3587–3602, 2017.

[13] Bernhard Etzlinger, Florian Meyer, Henk Wymeersch, Franz Hlawatsch, Gerhard Müller, and Andreas Springer. Cooperative simultaneous localization and synchronization: Toward a low-cost hardware implementation. In 2014 IEEE 8th Sensor Array and Multichannel Signal Processing Workshop (SAM), pages 33–36. IEEE, 2014.

[14] Bernhard Etzlinger, Henk Wymeersch, et al. Synchronization and localization in wireless networks. Foundations and Trends® in Signal Processing, 12(1):1–106, 2018.

[15] Lorenzo Galleani and Patrizia Tavella. Time and the kalman filter. IEEE Control Systems Magazine, 30(2):44–65, 2010.

[16] X. Ge, Q. L. Han, X. M. Zhang, L. Ding, and F. Yang. Distributed event-triggered estimation over sensor networks: A survey. IEEE Transactions on Cybernetics, 50(3):1306–1320, 2020.

[17] G. Giorgi and C. Narduzzi. Performance analysis of kalman-filter-based clock synchronization in ieee 1588 networks. IEEE Transactions on Instrumentation and Measurement, 60(8):2902–2909, 2011.

[18] Michael Hamer and Raffaello D'Andrea. Self-calibrating ultra-wideband network supporting multi-robot localization. IEEE Access, 2018.

[19] Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In Signal processing, sensor fusion, and target recognition VI, volume 3068, pages 182–193. International Society for Optics and Photonics, 1997.

[20] Li Li, Dongdong Yu, Yuanqing Xia, and Hongjiu Yang. Event-triggered ukf for nonlinear dynamic systems with packet dropout. International Journal of Robust and Nonlinear Control, 27(18):4208–4226, 2017.

[21] Ian Loefgren, Nisar Ahmed, Eric Frew, Christoffer Heckman, and Sean Humbert. Scalable event-triggered data fusion for autonomous cooperative swarm localization. In 2019 22th International Conference on Information Fusion (FUSION), pages 1–8. IEEE, 2019.

[22] DecaWave Ltd. Application note: The implementation of two-way ranging with the dw1000 (aps013). IEEE Transactions on Aerospace and Electronic Systems, 2014.

[23] Pratap Misra and Per Enge. Global positioning system: Signals. Measurements and Performance,, pages 381–384, 2006.

[24] Luis E Navarro-Serment, Christiaan JJ Paredis, Pradeep K Khosla, et al. A beacon system for the localization of distributed robotic teams. In Proceedings of the International Conference on Field and Service Robotics, volume 6, pages 1–6, 1999.

[25] Dries Neirynck, Eric Luk, and Michael McLaughlin. An alternative double-sided two-way ranging method. In 2016 13th workshop on positioning, navigation and communications (WPNC), pages 1–4. IEEE, 2016.

[26] Michael Ouimet, David Iglesias, Nisar Ahmed, and Sonia Martínez. Cooperative robot localization using event-triggered estimation. Journal of Aerospace Information Systems, 15(7):427–449, 2018.

[27] R. T. Rajan and A. van der Veen. Joint ranging and clock synchronization for a wireless network. In 2011 4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), pages 297–300, 2011.

[28] D. Shi, T. Chen, and L. Shi. On set-valued kalman filtering and its application to event-based state estimation. IEEE Transactions on Automatic Control, 60(5):1275–1290, 2015.

[29] Dawei Shi, Tongwen Chen, and Ling Shi. An event-triggered approach to state estimation with multiple point-and set-valued measurements. Automatica, 50(6):1641–1648, 2014.

[30] J. Sidorenko, V. Schatz, N. Scherer-Negenborn, M. Arens, and U. Hugentobler. Decawave ultra-wideband warm-up error correction. IEEE Transactions on Aerospace and Electronic Systems, 57(1):751–760, 2021.

[31] J. Tiemann, Y. Elmasry, L. Koring, and C. Wietfeld. Atlas fast: Fast and simple scheduled tdoa for reliable ultra-wideband localization. In 2019 International Conference on Robotics and Automation (ICRA), pages 2554–2560, 2019.

[32] Jeffrey K Uhlmann. General data fusion for estimates with unknown cross covariances. In Signal Processing, Sensor Fusion, and Target Recognition V, volume 2755, pages 536–547. International Society for Optics and Photonics, 1996.

[33] Jeffrey K. Uhlmann, Simon J. Julier, Behzad Kamgar-Parsi, Marco O. Lanzagorta, and Haw-Jye S. Shyu. NASA Mars rover: a testbed for evaluating applications of covariance intersection. In Grant R. Gerhart, Robert W. Gunderson, and Chuck M. Shoemaker, editors, Unmanned Ground Vehicle Technology, volume 3693, pages 140 – 149. International Society for Optics and Photonics, SPIE, 1999.

[34] Weijie Yuan, Nan Wu, Bernhard Etzlinger, Hua Wang, and Jingming Kuang. Cooperative joint localization and clock synchronization based on gaussian message passing in asynchronous wireless networks. IEEE Transactions on Vehicular Technology, 65(9):7258–7273, 2016.