Approximate Distributed Kalman Filtering for Cooperative Multi-agent Localization

Wm. Joshua Russell
Electrical and Computer Engineering
University of California, Santa Barbara, USA

Prabir Barooah
Department of Mechanical and Aerospace Engineering
University of Florida, USA

Daniel J. Klein and João P. Hespanha
Department of Electrical and Computer Engineering
University of California, Santa Barbara, USA

1 Introduction

Target tracking typically refers to the problem of determining the position of a mobile agent based on a stream of noisy measurements. Here, we are interested in the problem of estimating the trajectory of a mobile agent based on noisy measurements collected by a team of autonomous vehicles — a problem that is relevant to applications such as surveillance, disaster relief, and scientific exploration.

Even small mobile agents typically have access to a multitude of sensing modalities capable of providing position information. When available, GPS provides information about the position of the agent carrying the GPS antenna in a earth-fixed coordinate system. Inertial Measurement Units (IMUs) or/and vision sensors (Borenstein et al., 1997; Nistér et al., 2004; Olson et al., 2003) can provide measurements of the relative position of an agent across time. However, since these measurements are noisy, their integration over time typically leads to high rates of error growth (Olson et al., 2003; Makadia & Daniilidis, 2005; Oskiper et al., 2007). Several sensors are also capable of providing measurements regarding relative positions between agents. These include vision-based sensors such as cameras and light detection and ranging sensors (LIDARs), or RF sensors using angle of arrival (AoA) and time difference of arrival (TDoA) measurements. The problem of fusing measurements regarding relative positions between mobile agents for estimating their locations is commonly known as cooperative localization in the robotics literature (Kurazume et al., 1994; Rekleitis et al., 2002; Mourikis & Roumeliotis, 2006).

The use of a group of mobile agents to track a mobile target can result in low estimation errors, even

This material is based upon work supported by the Institute for Collaborative Biotechnologies through grant W911NF-09-D-0001 from the U.S. Army Research Office. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

when based on measurements from fairly noisy sensors. This is especially so when the mobile agents are able to fuse information from a multitude of sensors to obtain accurate estimates of their own positions as well as the position of the target. The typical approach for cooperative localization is to use an extended Kalman filter, to fuse both odometry data and robot-to-robot relative distance measurements (Mourikis & Roumeliotis, 2006; Roumeliotis & Bekey, 2002). However, computing the Kalman filter estimates requires all measurements to be available at central processing unit, which requires reliable solutions to the difficult problem of routing data though an ad-hoc network of mobile agents (Mueller et al., 2004). The desire to avoid this difficulty has drawn significant attention to the problem of distributed Kalman filtering (Spanos et al., 2005; Alriksson & Rantzer, 2006; Olfati-Saber, 2007; Carli et al., 2007). The papers (Alriksson & Rantzer, 2006; Alriksson & Rantzer, 2007) in particular present a distributed Kalman filtering techniques for multi-agent localization, while (Zhang & Martonosi, 2008) addresses cooperative localization in mobile sensor networks with intermittent communication, in which an agent updates its prediction based on the agents it encounters, but does not use inter-agent relative position measurements.

Under infinite prior covariance, the Kalman filter computes best linear unbiased estimates (BLUE) (Mendel, 1995). It turns out that, for the problem at hand, the BLUE has a convenient structure that can be described in terms of a graph consisting of nodes (agent locations) and edges (relative measurements). This structure can be exploited to distribute computation by using parallel iterative methods of solving linear equations. We have explored this in our earlier localization work for static sensor networks (Barooah & Hespanha, 2005). More recently, we have proposed an approximate distributed localization algorithm for groups of mobile agents that explore this same structure (Barooah et al., 2010). We used simulations to show that the estimates produced by this algorithm can be remarkably close to the centralized Kalman/BLUE estimates, even with a modest amount of computation and communication.

In this paper we show how one of the distributed algorithms proposed in (Barooah et al., 2010) can be extended to solve the problem of tracking a moving target with linear dynamics and provide conditions under which this algorithm converges. For square lattice graphs we provide explicit bounds on the convergence rate of the distributed algorithm.

The rest of the paper is organized as follows. Section 2 provides the necessary background information, including the algorithm proposed in (Barooah et al., 2010) for the distributed localization of a group of mobile agents. Section 3 describes how this algorithm can be extended to target tracking and Section 4 provides the main convergence results. A few concluding remarks are provided in Section 5.

2 Background

The following problem was considered in (Barooah et al., 2010): A group of η mobile agents need to estimate their own positions with respect to a geostationary coordinate frame. The noisy measurement of the displacement of Agent i obtained by its on-board sensors during the k-th time interval is denoted by $u_i(k)$, so that

$$u_i(k) = x_i(k+1) - x_i(k) + w_i(k), \tag{1}$$

where $x_i(k)$ is the position of Agent i at time k, and $\{w_i(k)\}$ is a zero-mean noise with the property that the expected value $E[w_i(k)w_{i'}(k')] = 0$ unless $i = i', k = k'$. We assume that certain pairs of agents, say i, i', can also obtain noisy inter-agent measurements

$$y_{ii'}(k) = x_i(k) - x_{i'}(k) + v_{ii'}(k), \tag{2}$$

where $v_{ii'}(k)$ is zero-mean measurement noise with expected value $E[v_{ii'}(k)v_{jj'}(k')] = 0$ unless $(ii') = (jj'), k = k'$. Two agents connected by an inter-agent measurement are called *neighbors*. Occasionally some of the nodes have noisy absolute position measurements (e.g., obtained from GPS) of the form

$$y_i(k) = x_i(k) + v_{ii'}(k), \tag{3}$$

The goal is to combine the agent-to-agent relative position measurements with agent displacement measurements to obtain estimates of their locations that are more accurate than what is possible from the agent displacement measurements alone (i.e. *dead reckoning*).

This problem can be associated with a measurement graph $\mathcal{G}(\bar{k}) = (\mathcal{V}(\bar{k}), \mathcal{E}(\bar{k}))$ constructed as follows

1. The positions $x_i(k)$ of each agent $i \in \{1, \ldots, \eta\}$ at each time step $k \in \{0, 1, \ldots, \bar{k}\}$ is associated with one node in $\mathcal{V}(\bar{k})$ and an additional "reference" node x_0 that is associated with the origin of an inertial coordinate system is included. Note that the reference node does not generally correspond to an agent.

2. Each measurement is associated with a graph edge: agent displacement measurements (1) are associated with edges between nodes corresponding to the same agent at different time steps; inter-agent measurements (2) are associated with edges between nodes corresponding to different agents at the same time step; and absolute displacement measurements (3) are associated with edges between an agent at a given time instant and the reference node.

   All measurements mentioned above are of the type

$$\tilde{z}_e = x_i - x_{i'} + \varepsilon_e \tag{4}$$

where i and i' are node indices for the measurement graph $\mathcal{G}(\bar{k})$ and $e = (i, i')$ is an edge (an ordered pair of nodes) between nodes. In particular, an edge exists between a node i and node i' if and only if a relative measurement of the form (4) is available between the two nodes.

Since a measurement of $x_i - x_{i'}$ is different from that of $x_{i'} - x_i$, the edges in the measurement graph are directed, but the edge directions are arbitrary and simply have to be consistent with (4). Figure 1 shows an example of a measurement graph.

2.1 Centralized BLUE

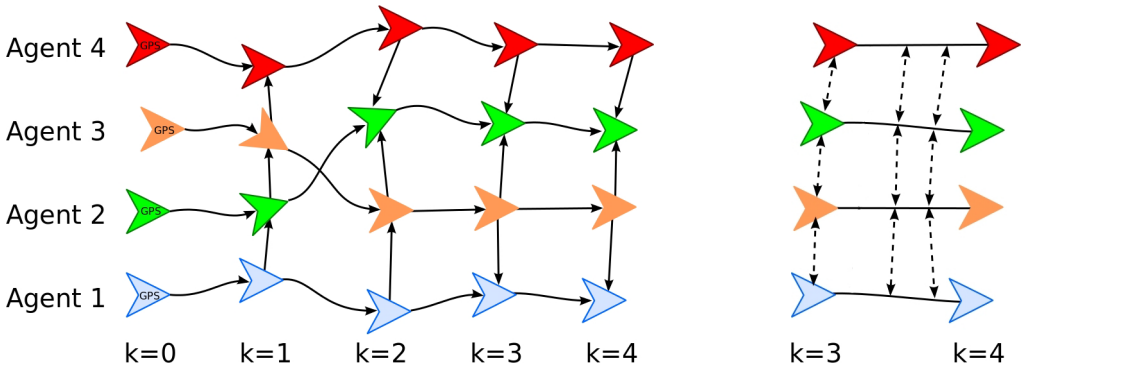
The best linear unbiased estimator (BLUE) has the smallest variance among all linear unbiased estimators. Consider again the measurement graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the nodes in $\mathcal{V}$ correspond to variables and edges in $\mathcal{E}$ correspond to relative measurement of the form (4). Let $\mathcal{V}_\mathcal{R}$ denote the reference node and $n = |\mathcal{V} \setminus \mathcal{V}_\mathcal{R}|$ be the number of unknown variables that are to be estimated.

Denote by $\mathbf{x}$, the vector obtained by stacking together the unknown variables, by $\mathbf{x}_\mathcal{R}$, the vector obtained by stacking together the reference node's state, by $\tilde{\mathbf{z}}$, the vector obtained by stacking together measurements $u_i(k)$ and $y_{ii'}(k)$, by ε, the vector obtained by stacking together noises $w_i(k)$ and $v_{ii'}(k)$, and by P, the matrix of error covariance. The system of relative measurements associated with the graph can then be written as

$$\tilde{\mathbf{z}} = B^\top \mathbf{x} + B_\mathcal{R}^\top \mathbf{x}_\mathcal{R} + \varepsilon, \tag{5}$$

where $[B^\top B_\mathcal{R}^\top]^\top$ is a generalized incidence matrix for $\mathcal{G}$. As described in (Barooah & Hespanha, 2007), given a measurement graph with $\bar{n}$ unknown variables, the BLUE, $\hat{\mathbf{x}}^*$ is given by the solution of a system of linear equations

$$\mathcal{L}\mathbf{x} = \mathbf{b}, \tag{6}$$



(a) Measurement graph $\mathcal{G}(4)$ (b) Communication between $k = 3$ and $k = 4$.

Figure 1: (a) An example of a measurement graph generated as a result of the motion of a group of four mobile agents. The graph shown here is $\mathcal{G}(4)$, i.e., the snapshot at the 4^{th} time instant. The unknown variables at current time $k = 4$ are the positions $x_i(k)$, $i \in \{1,\ldots,4\}$, at the time instants $k \in \{1,\ldots,4\}$. (b) The communication during the time interval between $k = 3$ and $k = 4$. In the situations shown in the figure, 4 rounds of communication occur between a pair of agents in this time interval.

where $\mathcal{L} = BP^{-1}B^\top$ and $\mathbf{b} = BP^{-1}(\tilde{\mathbf{z}} - B_{\mathcal{R}}^\top \mathbf{x}_{\mathcal{R}})$. The matrix $\mathcal{L}$ is invertible (so that the BLUE, $\hat{\mathbf{x}}^*$, exists and is unique) if and only if for every node, there is an undirected path between the node and the reference node (Barooah & Hespanha, 2005). Under this condition, the covariance matrix of the estimation error $\Sigma := \mathrm{Cov}(\hat{\mathbf{x}}^*, \hat{\mathbf{x}}^*)$ is given by

$$\Sigma = \mathcal{L}^{-1}.$$

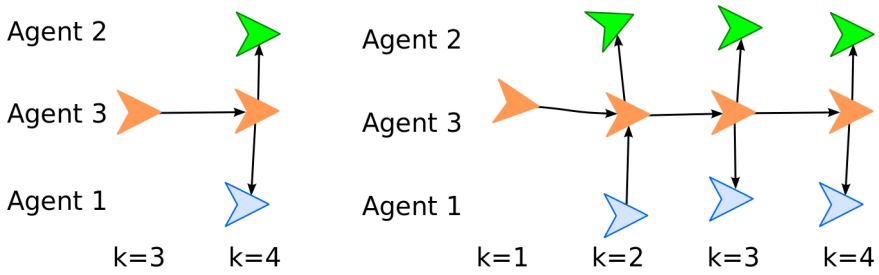
If all of the measurements corresponding to the edges in $\mathcal{G}(\bar{k})$ are available to a central processor at time $\bar{k}$, the processor can compute the BLUE of all the node variables in the graph $\mathcal{G}(\bar{k})$ (which correspond to the present as well as past positions of the agents) by solving (6).

2.2 A Distributed Algorithm for Dynamic Localization

In this section we describe the iterative distributed algorithm proposed in (Barooah et al., 2010, Algorithm A.2) to obtain estimates of the mobile agents' positions that are close to the centralized BLUE algorithm described above. To ease later discussion, we call this the *Mobile Agent Iterative Algorithm* (or MAI algorithm).

In the description below, κ is a fixed positive integer that denotes the "size" of a subgraph of $\mathcal{G}(\bar{k})$ that every agent maintains in its local memory at time $\bar{k}$. The parameter κ is essentially fixed ahead of time and provided to all agents; its value is determined by the memory available to each agent's local processor. The maximum number of iterations carried out by an agent during the interval between times $\bar{k}$ and $\bar{k}+1$, is denoted by $\bar{\tau}$ and depends on the maximum number of communication rounds between Agent i and its neighbors during this interval.

Let $\mathcal{G}(\bar{k})^\kappa = (\mathcal{V}(\bar{k})^\kappa, \mathcal{E}(\bar{k})^\kappa)$ be the subgraph containing the nodes, $\mathcal{V}(\bar{k})^\kappa$, associated with all agent states from time $\max(\bar{k}-\kappa, 0)$ until time $\bar{k}$ and containing edges, $\mathcal{E}(\bar{k})^\kappa$, corresponding to relative measurements between two nodes in $\mathcal{G}(\bar{k})^\kappa$. More simply, $\mathcal{G}(\bar{k})^\kappa$ is the subgraph containing all nodes and edges corresponding to states of agents and relative measurements at the current and previous κ time instants. In this case, $\hat{\mathbf{x}}_i^{(\tau)}(\bar{k})$ is a vector of Agent i's state estimates from time $\max(\bar{k}-\kappa, 0)$ to time $\bar{k}$, obtained in the



(a) $\kappa = 1$ (b) $\kappa = 3$

Figure 2: Truncated subgraphs, $\mathcal{G}_3(4)$ of Agent 3 at time 4 for the measurement graph shown in Figure 1.

τ^{th} iteration. During the interval between time indices $\bar{k}$ and $\bar{k}+1$, each agent, i updates the estimate of $\mathbf{x}_i(\bar{k})$ in the following way.

1. initialization: $\hat{x}_i^{(0)}(\bar{k}) = \hat{x}_i^{(\bar{\tau})}(\bar{k}-1) + u_i(\bar{k}-1)$.
2. collect inter-agent measurements that occurred at time $\bar{k}$, i.e., obtain $y_{ii'}(\bar{k})$ for neighbor agents, i'.
3. iterative update: Agent i treats its neighbors estimates, as well as its oldest estimate as reference variables and computes the BLUE estimate for its own states within the subgraph $\mathcal{G}_i(\bar{k})^{\kappa}$.

Because each agent treats its oldest estimate as a reference, these estimates are fixed and cannot change. Due to truncation, older measurements are no longer used in computing estimates but, the information that is contained in these measurements still has an effect on the current estimates (a la Kalman filtering). The penalty for truncating the measurement graph prior to the BLUE being reached is that some error is fixed into future estimates. In other words, at some point new measurements do not change old estimates. Later we show that even with a short history of data, this error tends to be small.

The MAI algorithm continues as long as the agents continue to move. Figure 2 shows an example of the local subgraphs used by an agent (Agent 3 in Figure 1) for two cases, $\kappa = 1$ and $\kappa = 3$. Note that the proposed algorithm is particularly simple when $\kappa = 1$, since in that case the iterative update is simply the solution to the following equation:

$$S_i(\bar{k}-1)\hat{x}_i^{\tau}(\bar{k}) = Q_i^{-1}(\bar{k}-1)\Big(\hat{x}_i^{\bar{\tau}}(\bar{k}-1) + u_i(\bar{k}-1)\Big) + \sum_{i' \in \mathcal{N}_i(\bar{k})} R_{ii'}^{-1}(\bar{k})\Big(\hat{x}_{i'}^{\tau-1}(\bar{k}) + y_{ii'}(\bar{k})\Big),$$

where $Q_i(\bar{k}) := \text{Cov}(w_i(\bar{k}), w_i^{\top}(\bar{k}))$ and $R_{ii'}(\bar{k}) := \text{Cov}(v_{ii'}(\bar{k}), v_{ii'}^{\top}(\bar{k}))$ are the error covariances in the displacement measurement $u_i(\bar{k}-1)$ and inter-agent relative measurement $y_{ii'}(\bar{k})$, respectively, and $S_i(\bar{k}) := Q_i^{-1}(\bar{k}) + \sum_{i' \in \mathcal{N}_i(\bar{k})} R_{ii'}^{-1}(\bar{k})$. When all the measurement error covariances are equal, the update is simply:

$$\hat{x}_i^{\tau}(\bar{k}) = \frac{1}{|\,\mathcal{N}_i(\bar{k}) + 1\,|}\Big(\hat{x}_i^{\bar{\tau}}(\bar{k}-1) + u_i(\bar{k}-1) + \sum_{i' \in \mathcal{N}_i(\bar{k})} (\hat{x}_i^{\prime\tau-1}(\bar{k}) + y_{ii'}(\bar{k}))\Big),$$

where $\mathcal{N}_i(\bar{k})$ denotes the set of Agent i's neighbors at time $\bar{k}$.

When $\kappa = \infty$, the MAI algorithm is simply the Jacobi iterations to compute the BLUE estimates of all the node variables in the graph $\mathcal{G}(\bar{k})$, i.e., the past and present states of the agents. In this case, (Barooah

et al., 2010, Proposition 1) guarantees that the estimates computed converge to the BLUE estimates as $\bar{\tau} \to \infty$. When the MAI algorithm is implemented with small values of κ, after a certain number of time steps, measurements from times earlier than κ steps into the past are no longer directly used. However, the information from past measurements is still used indirectly, since it affect the values of the reference variables used by the agents for their local subgraphs.

With finite κ, the estimates typically do not reach their centralized optimal because, while the measurement before time $k-\kappa$ are not completely forgotten (as explained above) they are not incorporated in the optimal fashion. A further reduction in accuracy comes from the fact that in practice $\bar{\tau}$ may not be large enough to get close to convergence even in the truncated local subgraphs. The MAI algorithm therefore produces an approximation of the centralized optimal estimates; the approximation becomes more accurate as $\bar{\tau}$ and κ increase.

2.2.1 Communication and Computation Costs

The amount of data an agent needs to store and broadcast increases as the "size" of the truncated local subgraph that the agent keeps in local memory increases, and therefore, on κ. When the neighbors of Agent i do not change with time, the number of nodes in the local truncated subgraph of an agent at any given time is $\kappa + \mathcal{N}_i\kappa$, where $\mathcal{N}_i$ is the number of neighbors of the Agent i. In this case, the number of edges in the truncated local subgraph is at most $\kappa + \kappa\mathcal{N}_i$ (the first term is the number of inertial measurements and the second term is the number of relative measurements between the agent and its neighboring agents that appear as edges in the subgraph). Therefore, an Agent i needs a local memory large enough to store $d[2\kappa(1+\mathcal{N}_i) + \kappa\mathcal{N}_i]$ floating-point numbers, where $d = 2$ or 3 depending on whether positions are 2 or 3 dimensional vectors. An agent has to broadcast the current estimates of its interior node variables, i.e., $d\kappa$ numbers, at every iteration. Thus, the communication, computation and memory requirements of the MAI algorithm are quite low for small values of κ. It is assumed that the agents can obtain the error covariances of the measurements on the edges that are incident on themselves, so these need not be transmitted.

3 Target Tracking

The problem presented in Section 2 assumed that all agents compute temporal displacement measurements. In some cases, such as target tracking, there may be agents external to the network, that do not provide such measurements to the network. We refer to such agents as *target agents* and the remaining agents will be called *cooperative agents*. In this section we assume that our system includes a target agent. We show how to incorporate target agents with linear dynamics into the estimation framework introduced above.

3.1 Extension to General Linear Dynamics

Recall from the distributed algorithms presented in Section 2.2 that, at each iteration, each agent computes the BLUE of its local subgraph. Previously, only absolute and relative position measurements were considered, for which the local BLUE is solvable if every node in the subgraph is connected to at least one reference node.

Consider now a single target agent with with state $\mathbf{x}_{\mathrm{T}}(k)$, that moves according to the linear model

$$\mathbf{x}_{\mathrm{T}}(k+1) = F(k)\mathbf{x}_{\mathrm{T}}(k) + G(k)\mathbf{u}_{\mathrm{T}}(k) + \mathbf{w}_{\mathrm{T}}(k), \tag{7}$$

where $\mathrm{Cov}(\mathbf{w}_{\mathrm{T}}(k), \mathbf{w}_{\mathrm{T}}(k)) = Q_{\mathrm{T}}(k)$ is the target model noise. We now allow the target's state to be multi-dimensional (i.e. position and velocity).

Along with the target model, measurements of the target are made relative to a reference state, $\mathbf{x}_{\mathcal{R}}(k)$,

$$\mathbf{y}_{\mathrm{T}}(k) = H_{\mathcal{R}}(k)\mathbf{x}_{\mathcal{R}}(k) + H_{\mathrm{T}}(k)\mathbf{x}_{\mathrm{T}}(k) + \mathbf{v}_{\mathrm{T}}(k). \tag{8}$$

Here $\mathrm{Cov}(\mathbf{v}_{\mathrm{T}}(k), \mathbf{v}_{\mathrm{T}}(k)) = R_{\mathrm{T}}(k)$ is observation noise and $H(k) = [H_{\mathcal{R}}(k)\ H_{\mathrm{T}}(k)]$ is an observation matrix that is appropriately defined so entries of $\mathbf{y}_{\mathrm{T}}(k)$ are the measurements of the target's position relative to the reference state. The noise vectors, $\mathbf{w}_{\mathrm{T}}(k)$ and $\mathbf{v}_{\mathrm{T}}(k)$ are assumed to be uncorrelated with time, uncorrelated with the network's state, and uncorrelated with the *model input* vector, $\mathbf{u}_{\mathrm{T}}(k)$.

As was done previously, we rearrange the model equations (7), to make this problem fit the relative measurement framework

$$G(k)\mathbf{u}_{\mathrm{T}}(k) = \mathbf{x}_{\mathrm{T}}(k+1) - F(k)\mathbf{x}_{\mathrm{T}}(k) - \mathbf{w}_{\mathrm{T}}(k).$$

Also, subtracting the reference state from both sides of the measurements in (8) results in

$$\mathbf{y}_{\mathrm{T}}(k) - H_{\mathcal{R}}(k)\mathbf{x}_{\mathcal{R}}(k) = H_{\mathrm{T}}(k)\mathbf{x}_{\mathrm{T}}(k) + \mathbf{v}_{\mathrm{T}}(k).$$

We can now stack the target measurements and modeled dynamics into a single vector

$$\begin{bmatrix} \mathbf{y}_{\mathrm{T}}(1) \\ G(1)\mathbf{u}_{\mathrm{T}}(1) \\ \mathbf{y}_{\mathrm{T}}(2) \\ G(2)\mathbf{u}_{\mathrm{T}}(2) \\ \vdots \\ G(\bar{k}-1)\mathbf{u}_{\mathrm{T}}(\bar{k}-1) \\ \mathbf{y}_{\mathrm{T}}(\bar{k}) \end{bmatrix} - \begin{bmatrix} H_{\mathcal{R}}(1) & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & H_{\mathcal{R}}(2) & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & H_{\mathcal{R}}(\bar{k}) \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathcal{R}}(1) \\ \mathbf{x}_{\mathcal{R}}(2) \\ \vdots \\ \mathbf{x}_{\mathcal{R}}(\bar{k}-1) \\ \mathbf{x}_{\mathcal{R}}(\bar{k}) \end{bmatrix} =$$

$$\begin{bmatrix} H_{\mathrm{T}}(1) & 0 & \cdots & 0 & 0 \\ -F(1) & \mathrm{I} & \cdots & 0 & 0 \\ 0 & H_{\mathrm{T}}(2) & \cdots & 0 & 0 \\ 0 & -F(2) & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & -F(\bar{k}-1) & \mathrm{I} \\ 0 & 0 & \cdots & 0 & H_{\mathrm{T}}(\bar{k}) \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathrm{T}}(1) \\ \mathbf{x}_{\mathrm{T}}(2) \\ \vdots \\ \mathbf{x}_{\mathrm{T}}(\bar{k}-1) \\ \mathbf{x}_{\mathrm{T}}(\bar{k}) \end{bmatrix} + \begin{bmatrix} \mathbf{v}_{\mathrm{T}}(1) \\ -\mathbf{w}_{\mathrm{T}}(1) \\ \mathbf{v}_{\mathrm{T}}(2) \\ -\mathbf{w}_{\mathrm{T}}(2) \\ \vdots \\ -\mathbf{w}_{\mathrm{T}}(\bar{k}-1) \\ \mathbf{v}_{\mathrm{T}}(\bar{k}) \end{bmatrix}.$$

For simplicity, this equation is rewritten as

$$\tilde{\mathbf{z}}_{\mathrm{T}} - H_{\mathcal{R}}\mathbf{x}_{\mathcal{R}} = H_{\mathrm{T}}\mathbf{x}_{\mathrm{T}} + \boldsymbol{\varepsilon}_{\mathrm{T}},$$

where $\mathbf{x}_{\mathrm{T}}$ denotes the target state for all $k \in \{1, 2, \ldots, \bar{k}\}$, $\tilde{\mathbf{z}}_{\mathrm{T}}$ denotes measurements of linear combinations of the target state, and $\boldsymbol{\varepsilon}_{\mathrm{T}}$ denotes the noise associated with these measurements. Denote by P_{T} the covariance of $\boldsymbol{\varepsilon}_{\mathrm{T}}$.

Following the conventions established earlier, the BLUE of target state given $\tilde{\mathbf{z}}_{\mathrm{T}}$ is defined to be

$$\hat{\mathbf{x}}_{\mathrm{T}}^{*} \equiv (H_{\mathrm{T}}P_{\mathrm{T}}^{-1}H_{\mathrm{T}}^{\top})^{-1}H_{\mathrm{T}}P_{\mathrm{T}}^{-1}(\tilde{\mathbf{z}}_{\mathrm{T}} - H_{\mathcal{R}}\mathbf{x}_{\mathcal{R}}). \tag{9}$$

For the BLUE to be feasible, conditions for which (9) has a solution must be established.

Theorem 1. *[Solvability of the BLUE with Modeled Dynamics]*
The BLUE, (9)*, has a unique solution if and only if the stochastic linear system given by* (7) *and* (8) *is observable at each time k.*

Proof. For the BLUE to exist $H_T P_T^{-1} H_T^\top$ must be invertible. For $\bar{k} > 1$, H_T has more rows than columns. Furthermore, P_T is a symmetric positive definite matrix. Therefore, $H_T P_T^{-1} H_T^\top$ is invertible if and only if H_T is full column rank. Here we show that this is the case if and only if the linear system described by (7) and (8) is observable at time $\bar{k}$.

First, it is shown that if H_T is full column rank, then the system is observable at time $\bar{k}$. Assume for contra-position that the system is not observable at time $\bar{k}$, then there exists a vector $\mathbf{x}_T(1) \neq 0$ such that

$$\begin{bmatrix} H_T(1) \\ H_T(2)F(1) \\ \vdots \\ H_T(\bar{k})F(\bar{k}-1)\dots F(1) \end{bmatrix} \mathbf{x}_T(1) = 0.$$

Furthermore,

$$H_T \begin{bmatrix} \mathrm{I} \\ F(1) \\ \vdots \\ F(\bar{k}-2)F(\bar{k}-3)\dots F(1) \\ F(\bar{k}-1)F(\bar{k}-2)\dots F(1) \end{bmatrix} \mathbf{x}_T(1) = \begin{bmatrix} H_T(1) \\ 0 \\ H_T(2)F(1) \\ \vdots \\ 0 \\ H_T(\bar{k})F(\bar{k}-1)\dots F(1) \end{bmatrix} \mathbf{x}_T(1) = 0.$$

As we have multiplied H_T by a non-trivial matrix and the result was zero, H_T is not full rank. Contra-position leads to the conclusion that if H_T is full rank then (7) and (8) is observable at time $\bar{k}$.

Now, we show that if the system in (7) and (8) is observable at time $\bar{k}$ then H_T is full column rank. Assume for contradiction that the system is observable at time k and H_T is not full column rank. Because H_T is not full column rank, there exists vector, $\mathbf{x}_T = [\mathbf{x}_T(1)^\top \mathbf{x}_T(2)^\top \dots \mathbf{x}_T(\bar{k})^\top]^\top$, such that $H_T \mathbf{x}_T = 0$. Expanding on this,

$$H_T \mathbf{x}_T = \begin{bmatrix} H_T(1)\mathbf{x}_T(1) \\ \mathbf{x}_T(2) - F(1)\mathbf{x}_T(1) \\ \vdots \\ \mathbf{x}_T(\bar{k}) - F(k-1)\mathbf{x}_T(\bar{k}-1) \\ H(\bar{k}) \end{bmatrix} = 0.$$

Consider only even rows of $H_T \mathbf{x}_T$, for $k \in \{2, 3, \dots, \bar{k}\}$

$$x_T(k) = F(k-1)x_T(k-1) = F(k-1)F(k-2)\dots F(1)x_T(1). \tag{10}$$

Substitution (10) into the odd rows of $H_T \mathbf{x}_T$,

$$\begin{bmatrix} H_T(1)\mathbf{x}_T(1) \\ H_T(2)\mathbf{x}_T(2) \\ \vdots \\ H_T(\bar{k})\mathbf{x}_T(\bar{k}) \end{bmatrix} = \begin{bmatrix} H_T(1)\mathbf{x}_T(1) \\ H_T(2)F(1)\mathbf{x}_T(1) \\ \vdots \\ H_T(\bar{k})F(\bar{k}-1)\dots F(1)\mathbf{x}_T(1) \end{bmatrix} = \mathcal{O}(\bar{k})\mathbf{x}_T(1) = 0,$$

where $\mathcal{O}(\bar{k})$ is the observability matrix at time $\bar{k}$. This contradicts the assumption that (7) and (8) is observable at time $\bar{k}$. □

Note that the inclusion of modeled dynamics can be extended from target agents to all agents. So long as the measurements and models of each agents local BLUE are observable, the local BLUEs are solvable and the larger iterative Jacobi procedure is tractable.

3.2 General Partitioning of the Measurement Graph

In the target tracking problem, it is assumed that the target agent is non-cooperative. Because of this, a cooperative agent must take responsibility for estimating the target's state, we call that agent the *tracking agent*. One naïve way to accomplish this is to have the tracking agent perform two optimizations per iteration, one for itself and one for the target. This would work, but there is a simpler method that speeds convergence.

In Section 2.1 we used a graph to represent the distributed estimation problem. In Section 2.2 we partitioned the graph's vertex set into η independent subsets $\mathcal{V}_i^{\text{int}}, i \in \{1,2,\dots\eta\}$, one for each agent. Each agent then considered only a subgraph consisting of their nodes, all neighboring nodes, and the edges connecting said nodes. To compute the BLUE of the system's parameters, we employed an iterative Jacobi algorithm in which each agent received their neighbors state estimates and computed the BLUE of their local subgraph assuming their neighbors estimates were noiseless. In this case we employed a partitioning of the vertex set that was intuitive because each subset contained nodes belonging to a single agent.

Now, we show that the convergence results for distributed algorithms in Section 2.2 are preserved under an arbitrary partitioning of the vertex set. Assume that the vertex set $\mathcal{V}$ is partitioned into μ independent subsets, $\mathcal{V}_i, i \in \{1,2,\dots\mu\}$. To analyze the block-Jacobi method, we first index the nodes of the graph so that nodes belonging to a given partition are indexed sequentially. With this indexing, the system of equations in (5) can be written

$$\tilde{\mathbf{z}} - B_{\mathcal{R}}^{\top}\mathbf{x}_{\mathcal{R}} = \begin{bmatrix} B_1^{\top} & B_2^{\top} & \dots & B_{\mu}^{\top} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{\mu} \end{bmatrix} + \boldsymbol{\varepsilon},$$

where the vector $\mathbf{x}_i$ contains all of the node variables in the partition i, $B = \begin{bmatrix} B_1^{\top} & B_2^{\top} & \dots & B_{\mu}^{\top} \end{bmatrix}^{\top}$ denotes the generalized incidence matrix for the unknown states, and $\boldsymbol{\varepsilon}$ the noise vector. The generalized Laplacian matrix, $\mathcal{L}$, is a block matrix defined to be

$$\mathcal{L} \equiv BP^{-1}B^{\top} = \begin{bmatrix} B_1P^{-1}B_1^{\top} & B_1P^{-1}B_2^{\top} & \dots & B_1P^{-1}B_{\mu}^{\top} \\ B_2P^{-1}B_1^{\top} & B_2P^{-1}B_2^{\top} & \dots & B_2P^{-1}B_{\mu}^{\top} \\ \vdots & \vdots & \ddots & \vdots \\ B_{\mu}P^{-1}B_1^{\top} & B_{\mu}P^{-1}B_2^{\top} & \dots & B_{\mu}P^{-1}B_{\mu}^{\top} \end{bmatrix},$$

where P denotes the covariance of $\boldsymbol{\varepsilon}$. The generalized Laplacian matrix can be split into $\mathcal{L} = D - A$, where the generalized degree matrix is

$$D = \begin{bmatrix} B_1P^{-1}B_1^{\top} & 0 & \dots & 0 \\ 0 & B_2P^{-1}B_2^{\top} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & B_{\mu}P^{-1}B_{\mu}^{\top} \end{bmatrix},$$

and the generalized adjacency matrix is

$$A = \begin{bmatrix} 0 & -B_1P^{-1}B_2^{\top} & \dots & -B_1P^{-1}B_{\mu}^{\top} \\ -B_2P^{-1}B_1^{\top} & 0 & \dots & -B_2P^{-1}B_{\mu}^{\top} \\ \vdots & \vdots & \ddots & \vdots \\ -B_{\mu}P^{-1}B_1^{\top} & -B_{\mu}P^{-1}B_2^{\top} & \dots & 0 \end{bmatrix}.$$

As previously discussed, the BLUE is the unique solution for (6), which can be rewritten as $D\mathbf{x} = A\mathbf{x} + \mathbf{b}$, leading to the following block-Jacobi iterative method for solving it:

$$\hat{\mathbf{x}}(\tau+1) = D^{-1}A\hat{\mathbf{x}}(\tau) + D^{-1}\mathbf{b}, \tag{11}$$

where A is a sparse matrix . In particular, the $\{i, i'\}$ block element of A is non-zero if and only if $\mathcal{V}_i$ and $\mathcal{V}_{i'}$ contain nodes that are neighbors in $\mathcal{G}$ (i.e. $B_i B_{i'}^\top \neq 0$). This sparsity makes it possible to compute the updates in (11) in a distributed manner; each $\mathcal{V}_i$ only needs to communicate with its neighboring graph partitions. The following theorem is a convergence result for the block-Jacobi algorithm under arbitrary partitioning.

Theorem 2. *[Convergence of Block-Jacobi Iteration Under General Graph Partitioning]*
Given an arbitrary partitioning of the graph's node set, $\mathcal{V}$, if B is full row rank and $B_i B_i^\top$ is invertible for each partition, then the block-Jacobi algorithm, (11)*, converges.*

Proof. The block-Jacobi algorithm, (11), converges if and only if $\rho(D^{-1}A) < 1$. If B is full row rank, then $\mathcal{L}$ is positive definite. This comes about because P is positive definite (it is a covariance matrix) and $\mathcal{L}$ is an inner product of B weighted by P^{-1}. Further more, if $B_i B_i^\top$ is invertible for every partition, then each diagonal block of D is positive definite, rendering D positive definite. Finally, with $\mathcal{L}$ and D positive definite, the following inequalities hold

$$\begin{aligned} \mathcal{L} = D - A > 0 \\ \rho(D^{-1}\mathcal{L}) = \rho(I - D^{-1}A) > 0 \\ 1 > \rho(D^{-1}A). \end{aligned}$$

□

It should be noted that when $B_i B_i^\top$ is invertible the BLUE for the i^{th} subgraph is computable. This condition is true when considering only relative difference measurements, if every node is connected to at least one reference node. More generally, this condition is true when the conditions in Theorem 1 are met.

From this result, it is evident that the vertex set for a tracking agent can be extended to include the nodes belonging to the target. The tracking agent then implements the the aforementioned distributed algorithm and simultaneously estimates its own state, as well as the states belonging to the target agent. By doing the computation concurrently the tracking agent reduces communication and convergence time for the algorithm.

The idea of partitioning the vertex set for use in distributed estimation has two extremes. In the most distributed case, each agent has only one node in its vertex set. This was presented in (Barooah & Hespanha, 2005) . In the least distributed case, there is no partitioning of the vertex set and state estimation is done in a centralized manner. The fact that any partitioning of the vertex set leads to an optimal result is useful because one can cluster nodes together on any level.

4 Analysis

In this section we provide analysis for the convergence rate of the MAI algorithm. First we consider the convergence properties of the algorithm with infinite memory (i.e. estimating all unknown states). Then we look at the convergence properties of the algorithm with finite memory. As with algorithms of this

sort, convergence rates are highly dependent on topology. As such, we consider a network that resembles a rectangular lattice. For a square lattice graph, and uniform measurement covariances, we provide tight bounds for the infinite memory case. For the case of finite memory, we show the trade-off between memory length and number of iterations at each step. Even under modest computational requirements (i.e. few iterations and a short memory length), the MAI algorithm provides estimates that are much better than non-cooperative estimates; commonly referred to as *dead reckoning*.

4.1 Convergence Rate on a Grid Network

Consider again a network of η agents that make d-dimensional synchronous inter-agent measurements. Furthermore, assume that the inter-agent measurements at a particular time make a chain graph and that the neighbor relations are time invariant. Suppose that the agent's positions at time $k = 0$ are known exactly from GPS but that the agents lose GPS and time elapses until some time, $\bar{k}$. The corresponding measurement graph is an η by $\bar{k}+1$ grid, rendering the graph shown in Figure 3.

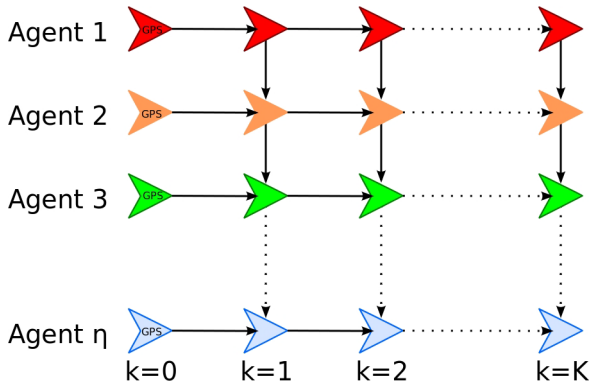


Figure 3: Hexagonal lattice graph corresponding to a chain of agents moving together in time.

To determine the convergence rate of the algorithm for agents with infinite memory, we must determine the convergence rate of the BLUE algorithm on the graph shown in Figure 3. To start, we reintroduce the displacement measurement of an agent given in (1). Denote by u_i the $d\bar{k}$-length vector of ordered displacement measurements for Agent i up to time $\bar{k}$,

$$\mathbf{u}_i = \left[\mathbf{u}_i(0)^\top, \mathbf{u}_i(1)^\top, \cdots, \mathbf{u}_i(\bar{k})^\top\right]^\top.$$

Similarly, denote by $\mathbf{y}_{ii'}$ the $d\bar{k}$-length vector of ordered relative position measurements between Agent i and Agent i' up to time $\bar{k}$,

$$\mathbf{y}_{ii'} = \left[\mathbf{y}_{ii'}(0)^\top, \mathbf{y}_{ii'}(1)^\top, \cdots, \mathbf{y}_{ii'}(\bar{k})^\top\right]^\top.$$

Denote by $\mathbf{x}_{\mathcal{R}i}$ the state of Agent i at $k = 0$ (the GPS fix) and denote by $\mathbf{x}_i$ the $d\bar{k}$-length vector of Agent i's unknown states ordered chronologically. Furthermore, set the notation for error vectors to match the notation for measurements; $\mathbf{u}_i$ has error $\mathbf{w}_i$ and $\mathbf{y}_{ii'}$ has error $\mathbf{v}_{ii'}$. Denote by Q_i and $R_{ii'}$ the covariance matrices of $\mathbf{w}_i$ and $\mathbf{v}_{ii'}$, respectively.

With these structures properly defined, the network of measurements can be written

$$\begin{bmatrix} \mathbf{u}_1 \\ \mathbf{y}_{12} \\ \mathbf{u}_2 \\ \mathbf{y}_{23} \\ \vdots \\ \mathbf{y}_{\eta\,\eta-1} \\ \mathbf{u}_\eta \end{bmatrix} = \begin{bmatrix} \mathbf{e}_1 & B_{\mathrm{I}}^\top & 0 & 0 & \cdots & 0 & 0 \\ 0 & -\mathrm{I}_{d\bar{k}} & 0 & \mathrm{I}_{d\bar{k}} & \cdots & 0 & 0 \\ 0 & 0 & \mathbf{e}_1 & B_{\mathrm{I}}^\top & \cdots & 0 & 0 \\ 0 & 0 & 0 & -\mathrm{I}_{d\bar{k}} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & \mathrm{I}_{d\bar{k}} \\ 0 & 0 & 0 & 0 & \cdots & \mathbf{e}_1 & B_{\mathrm{I}}^\top \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathcal{R}1} \\ \mathbf{x}_1 \\ \mathbf{x}_{\mathcal{R}2} \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{\mathcal{R}\eta} \\ \mathbf{x}_\eta \end{bmatrix} + \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{v}_{12} \\ \mathbf{w}_2 \\ \mathbf{v}_{23} \\ \vdots \\ \mathbf{v}_{\eta\,\eta-1} \\ \mathbf{w}_\eta \end{bmatrix}, \tag{12}$$

where $\mathbf{e}_1$ denotes the Kronecker product of length $\bar{k}$ elementary vector and the size d identity matrix, $\mathbf{e}_1 = [\mathrm{I}_d, 0_d, \cdots, 0_d]^\top$, 0 denotes the appropriately sized zero matrix, and B_{I} denotes the Kronecker product of the incidence matrix of an agent's non-GPS nodes and the size d identity matrix. From the problem formulation, an agent's nodes form a chain graph; B_{I} is a square matrix containing the last $d\bar{k}$ rows of an incidence matrix representing a chain graph. In general,

$$B_{\mathrm{I}}^\top = \begin{bmatrix} -\mathrm{I}_d & 0 & 0 & \cdots & 0 & 0 \\ \mathrm{I}_d & -\mathrm{I}_d & 0 & \cdots & 0 & 0 \\ 0 & \mathrm{I}_d & -\mathrm{I}_d & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -\mathrm{I}_d & 0 \\ 0 & 0 & 0 & \cdots & \mathrm{I}_d & -\mathrm{I}_d \end{bmatrix}.$$

For the special case when $\bar{k} = 1$, $B_{\mathrm{I}}^\top = [-\mathrm{I}_d]$.

4.1.1 The Best Linear Unbiased Estimate

The measurement vector, presented in (12) is a linear combination of reference states, $x_{\mathcal{R}i}$, and unknown states, x_i, corrupted by noise. The first step in computing the BLUE is to transform (12) into a linear system of unknown states. This is accomplished by subtracting the reference states from both sides of the measurement equation

$$\begin{bmatrix} \mathbf{u}_1 \\ \mathbf{y}_{12} \\ \mathbf{u}_2 \\ \mathbf{y}_{23} \\ \vdots \\ \mathbf{y}_{\eta\,\eta-1} \\ \mathbf{u}_\eta \end{bmatrix} - \begin{bmatrix} \mathbf{e}_1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ 0 & \mathbf{e}_1 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & \mathbf{e}_1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathcal{R}1} \\ \mathbf{x}_{\mathcal{R}2} \\ \vdots \\ \mathbf{x}_{\mathcal{R}\eta} \end{bmatrix} = \begin{bmatrix} B_{\mathrm{I}}^\top & 0 & \cdots & 0 \\ -\mathrm{I}_{d\bar{k}} & \mathrm{I}_{d\bar{k}} & \cdots & 0 \\ 0 & B_{\mathrm{I}}^\top & \cdots & 0 \\ 0 & -\mathrm{I}_{d\bar{k}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathrm{I}_{d\bar{k}} \\ 0 & 0 & \cdots & B_{\mathrm{I}}^\top \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_\eta \end{bmatrix} + \begin{bmatrix} \mathbf{w}_1 \\ \mathbf{v}_{12} \\ \mathbf{w}_2 \\ \mathbf{v}_{23} \\ \vdots \\ \mathbf{v}_{\eta\,\eta-1} \\ \mathbf{w}_\eta \end{bmatrix}. \tag{13}$$

To simplify this equation the following is introduced

$$\tilde{\mathbf{z}} = \begin{bmatrix} \mathbf{u}_1^\top & \mathbf{y}_{12}^\top & \mathbf{u}_2^\top & \mathbf{y}_{23}^\top & \cdots & \mathbf{y}_{\eta\,\eta-1}^\top & \mathbf{u}_\eta^\top \end{bmatrix}^\top,$$

$$\mathbf{x}_{\mathcal{R}} = \begin{bmatrix} \mathbf{x}_{\mathcal{R}1}^\top \mathbf{e}_1^\top & 0 & \mathbf{x}_{\mathcal{R}2}^\top \mathbf{e}_1^\top & 0 & \cdots & \mathbf{x}_{\mathcal{R}\eta}^\top \mathbf{e}_1^\top \end{bmatrix}^\top,$$

$$B = \begin{bmatrix} B_{\mathrm{I}} & -\mathrm{I}_{d\bar{k}} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \mathrm{I}_{d\bar{k}} & B_{\mathrm{I}} & -\mathrm{I}_{d\bar{k}} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \mathrm{I}_{d\bar{k}} & B_{\mathrm{I}} \end{bmatrix},$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1^\top & \mathbf{x}_2^\top & \cdots & \mathbf{x}_\eta^\top \end{bmatrix}^\top,$$

$$\varepsilon = \begin{bmatrix} \mathbf{w}_1^\top & \mathbf{v}_{12}^\top & \mathbf{w}_2^\top & \mathbf{v}_{23}^\top & \cdots & \mathbf{v}_{\eta\,\eta-1}^\top & \mathbf{w}_\eta \end{bmatrix}^\top,$$

$$P = \begin{bmatrix} Q_1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & R_{12} & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & Q_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & R_{23} & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & R_{\eta-1\,\eta} & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & Q_\eta \end{bmatrix}.$$

Now, (13) can be written

$$\tilde{\mathbf{z}} - \mathbf{x}_{\mathcal{R}} = B^\top \mathbf{x} + \varepsilon.$$

The BLUE, $\hat{x}^*$, of x is defined to be

$$\hat{\mathbf{x}}^* \equiv (BP^{-1}B^\top)^{-1} BP^{-1}(\tilde{\mathbf{z}} - \mathbf{x}_{\mathrm{R}}). \tag{14}$$

4.1.2 The Block-Jacobi Iterative Method

The algorithms presented in this paper use the block-Jacobi iterative method (introduced in Section 3.2) to compute the BLUE. We now examine the the block-Jacobi method as it applies to computing (14). The BLUE, $\hat{x}^*$, is the unique solution to the linear equation $\mathcal{L}\mathbf{x} = \mathbf{b}$. To implement the block-Jacobi method we split the matrix $\mathcal{L} = D - A$ so that D contains the block diagonal elements of $\mathcal{L}$. Then we add $A\mathbf{x}$ to both sides of the linear equation to form an estimate update law in accordance with the block-Jacobi method:

$$\hat{\mathbf{x}}(\tau+1) = D^{-1}A\hat{\mathbf{x}}(\tau) + D^{-1}\mathbf{b}. \tag{15}$$

Note that the update law is simply a discrete-time linear system. As such, convergence is determined solely by the eigenvalues of $D^{-1}A$.

To simplify further discussions, we define the *network iteration matrix* as

$$J \equiv D^{-1}A.$$

Denote by $\rho(J)$ the magnitude of the largest eigenvalue of J – commonly known as the spectral radius of J. Because J characterizes the dynamics of the Jacobi Algorithm, a discrete-time linear system, convergence of the algorithm is dependent on $\rho(J)$. It was shown in Theorem 2 that for this problem, the block-Jacobi method converges. We now show that it converges to the BLUE.

Theorem 3. *[Convergence to the BLUE]*
Given that the block-Jacobi iteration in (15) *converges,*

$$\lim_{\tau\to\infty} \hat{\mathbf{x}}(\tau) = \hat{\mathbf{x}}^*.$$

Proof. When (15) is convergent,

$$\hat{\mathbf{x}}(\infty) = D^{-1}A\hat{\mathbf{x}}(\infty) + D^{-1}\mathbf{b}.$$

From this

$$\begin{aligned} \hat{\mathbf{x}}(\infty) &= (D-A)^{-1}\mathbf{b} \\ &= (BP^{-1}B^\top)^{-1}BP^{-1}(\tilde{\mathbf{z}} - \mathbf{x}_{\mathcal{R}}) = \hat{\mathbf{x}}^*. \end{aligned}$$

□

Theorem 4. *[Rate of Convergence for Scalar Covariance Matrices]*
If the covariances, $P_{\{\cdot\}}$, for all measurements in the network are equal, then

$$\rho(J) \leq \frac{2}{4 - 2cos(\frac{\pi}{2k+1})}.$$

Proof. If all measurements have scalar covariance matrix, $P_s = \alpha \mathrm{I}$ for $\alpha \in \mathbb{R}$, the iteration matrix J can be written as

$$J = \begin{bmatrix} 0 & J_e & 0 & \cdots & 0 & 0 & 0 \\ J_c & 0 & J_c & \cdots & 0 & 0 & 0 \\ 0 & J_c & 0 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 0 & J_c & 0 \\ 0 & 0 & 0 & \cdots & J_c & 0 & J_c \\ 0 & 0 & 0 & \cdots & 0 & J_e & 0 \end{bmatrix},$$

where

$$\begin{aligned} J_e &= (B_I P_s^{-1} B_I^\top + P_s^{-1})^{-1} P_s^{-1} \\ &= (B_I B_I^\top + \mathrm{I})^{-1} \\ J_c &= (B_I P_s^{-1} B_I^\top + 2P_s^{-1})^{-1} P_s^{-1} \\ &= (B_I B_I^\top + 2\mathrm{I})^{-1}. \end{aligned}$$

From (Feingold & Varga, 1962), for at least one block row, i, each eigenvalue of J, $\lambda_k(J)$, satisfies

$$\left(\| (J_{i,i} - \lambda_k(J)\mathrm{I})^{-1} \|_p \right)^{-1} \leq \sum_{i' \neq i} \| J_{i,i'} \|_p. \tag{16}$$

Being that all of the diagonal blocks, $J_{i,i}$ are zero,

$$\left(\| (J_{i,i} - \lambda_k(J)\mathrm{I})^{-1} \|_p \right)^{-1} = \| \lambda_k(J)\mathrm{I} \|_p,$$

and (16) reduces to

$$\| \lambda_k(J)\mathrm{I} \|_p \leq \sum_{i' \neq i} \| J_{i,i'} \|_p.$$

Using the 2-norm in the previous inequality and taking the maximum sum over block rows,

$$\rho(J) \leq \sum_{i' \neq i} \| J_{i,i'} \|_2 = \max_i \sum_{i' \neq i} \rho(J_{i,i'}). \tag{17}$$

Equality in (17) comes from the fact that $\| J_{i,i'} \|$ is symmetric.

Careful inspection reveals that

$$B_I B_I^\top = \begin{bmatrix} 2 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 2 & -1 & \cdots & 0 & 0 \\ 0 & -1 & 2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & 2 & -1 \\ 0 & 0 & 0 & \cdots & -1 & 1 \end{bmatrix}.$$

$\bar{\tau}$ \ η	**1**	**2**	**3**	**4**
2	0	0	0	0
3	.0893	.0601	.0384	.0257
4	.0591	.0399	.0256	.0171
5	.0431	.0295	.0190	.0128

Table 1: For $\bar{\tau}$ iterations and η agents, the error of Upper Bounds Computed for $\rho(J)$

It was shown in (Trench, 1999), that the eigenvalues of $B_I B_I^\top$ are $\lambda_m = 2 - 2\cos\left(\frac{(2m-1)\pi}{2\bar{\tau}+1}\right)$ for $m \in \{1, 2, \cdots, \bar{\tau}\}$. Hence the eigenvalues of $(B_I B_I^\top + I)$ are $\lambda_m = 3 - 2\cos\left(\frac{(2m-1)\pi}{2\bar{\tau}+1}\right)$ for $m \in \{1, 2, \cdots, \bar{\tau}\}$. The smallest eigenvalue of $(B_I B_I^\top + I)$ is the inverse of the spectral radius of $J_e P_s^{-1}$, denoted $\rho_e = \frac{1}{3-2\cos\left(\frac{\pi}{2\bar{\tau}+1}\right)}$. In a similar manner, it can be shown that the spectral radius of $J_c P_s^{-1}$ is $\rho_c = \frac{1}{4-2\cos\left(\frac{\pi}{2\bar{\tau}+1}\right)}$.

Each block row of J features either one $J_c P_s^{-1}$ block or two $J_e P_s^{-1}$ blocks. By inspection, $\rho_e < 2\rho_c$, hence $\rho(J) \leq \frac{2}{4-2\cos\left(\frac{\pi}{2\bar{\tau}+1}\right)}$. □

There are a few things to note about this bound. First, the bound gets tighter as η gets large. Second, the bound is independent of the number of agents. Last, because $\frac{2}{4-2\cos\left(\frac{\pi}{2\bar{\tau}+1}\right)} < 1$, the Jacobi algorithm always converges.

To illustrate the tightness of the bound, the Table 1 shows the error of the upper bound for several values of $\bar{\tau}$ and η. Note that for $\eta > 2$ the bound becomes tighter as η gets larger.

4.2 Trade-off of Computation and Communication

Section 3.1 discussed how the distributed algorithm can be used to estimate the state of distributed dynamic systems. There are two main parameters affecting the performance of this algorithm, the amount of history retained for estimation, κ, and the number of iterations at each time step, $\bar{\tau}$. Here, performance refers to the covariance of estimate error. Higher performance relates to lower error covariance (i.e. estimates that are "tighter" to the optimal estimates). Both κ and $\bar{\tau}$ are positively correlated with performance; making these two values larger improves performance.

The first parameter, κ, is the amount of history considered when performing distributed estimation. When κ is finite it is no longer possible to obtain the global BLUE (the estimate that considers every measurement the network has ever produced). This is due to fixing the oldest estimates in the network. As a set of nodes become the oldest nodes in the graph, we deem them to be reference nodes and we take their corresponding position estimates as truth. By doing this, whatever error remains in the estimates is fixed forever. It will be demonstrated that even for κ relatively small, the method produces estimates that are near optimal and generally much better than dead reckoning. Furthermore, increasing κ produces diminishing returns. When increasing κ by a fixed amount, there is a larger performance increase when κ is small.

As an iterative algorithm, if the MAI algorithm is convergent then each iteration reduces the estimate error. As the number of iterations grows, the estimates converge asymptotically to the BLUE. As such, increasing $\bar{\tau}$ will improve the estimate's performance. As with κ, this effect has diminishing returns. When increasing $\bar{\tau}$ by a fixed amount, there is a larger performance increase when $\bar{\tau}$ is small. Also like κ, $\bar{\tau}$ is limited by constraints within the network. There is a finite limit to the number of iterations that are possible

(i.e. how much information can be shared) in a single time step. The density of the network and the bandwidth of the network's communication channels greatly affect how much information can be shared at each time step.

A logical question to ask when implementing the MAI algorithm on a dynamic system is just how well the algorithm will perform. This is not an easy question to answer as it depends not only on κ and $\bar{\tau}$, but on the network's topology and the measurement and model fidelities as well. A lower bound for the estimate error covariance is the estimate error covariance of the BLUE. Note that the BLUE is equivalent to the estimate one obtains from the MAI algorithm if $\kappa = \infty$ and $\bar{\tau} = \infty$. One can bound upper bound the estimate error covariance with that of dead-reckoning. For a given agent, the worst case scenario would be for each agent to estimate their states solely from inertial measurements. Under the assumption that the inertial measurements have independent error, one would determine an agent's current estimate error covariance by summing the covariances of inertial measurements from the agent's reference node to it's current node. Note that dead reckoning is equivalent to a MAI algorithm where $\bar{\tau} = 0$. In (Barooah & Hespanha, 2009) Barooah and Hespanha showed that adding a measurement to the network where there was not one previously can only reduce the error covariance of the BLUE. That is to say, by including inter-agent measurements in the estimation scheme can only serve to improve the resulting estimates.

Due to the topological dependence of the MAI algorithm, no general closed form solution for the estimate covariance can be provided. However, given a specific network it is possible to explicitly compute the estimate error covariances. To illustrate the utility of the MAI algorithm and also the trade-off between κ and $\bar{\tau}$, we present covariance results that were calculated for a scenario in which 10 agents are moving together. Initially, each agent knows it's position exactly. As the agents move along, they make relative position measurements to their nearest neighbors. As shown in Figure 3, the measurements graph evolves as a rectangular grid. The agents measurements (both inertial and inter-agent) are corrupted with error of unity covariance.

Figure 4 shows the trade-off between κ and $\bar{\tau}$. This figure depicts the agents' average error covariance norm of current state estimates after 50 time steps. With only one iteration per time step and only keeping two time steps in memory (the minimal parameters for the MAI algorithm to function), the agents' average error covariance norm is 5.86. For comparison, in an uncooperative environment (i.e. agents estimate states using dead reckoning) the average error covariance norm would be 50. The same metric for BLUE estimation (i.e. a centralized Kalman filter) is 5.55. Under very minimal assumptions, the algorithm provides estimates with performance close to that of Kalman filtering. If the algorithm parameters are increased so that $\kappa = 5$ and $\bar{\tau} = 5$, then the average error covariance norm shrinks to 5.59 a mere 0.7% higher than the that of Kalman filtering. Computationally, with these parameters the worst case is that an agent must solve a system of 12 equations and 4 unknowns 5 times per time step.

Figure 5 shows how the MAI algorithm performs as a smoother. This figure shows the agents' average error covariance norm for estimates at $k = 40$ using information gathered up to $k = 50$. These estimates approximate a fixed interval smoother. With only one iteration per time step and only keeping two time steps in memory, the agents' average error covariance norm is 4.85. For comparison, in an uncooperative environment the average error covariance norm would be 40. The same metric for BLUE estimation is 4.33. If the algorithm parameters are increased so that $\kappa = 5$ and $\bar{\tau} = 5$, then the average error covariance norm shrinks to 4.40 a mere 1.5% higher than the that of Kalman smoothing. Although these plots are for a particular scenario, the utility of the MAI algorithm is apparent. Under very reasonable conditions, the method performs quite well.

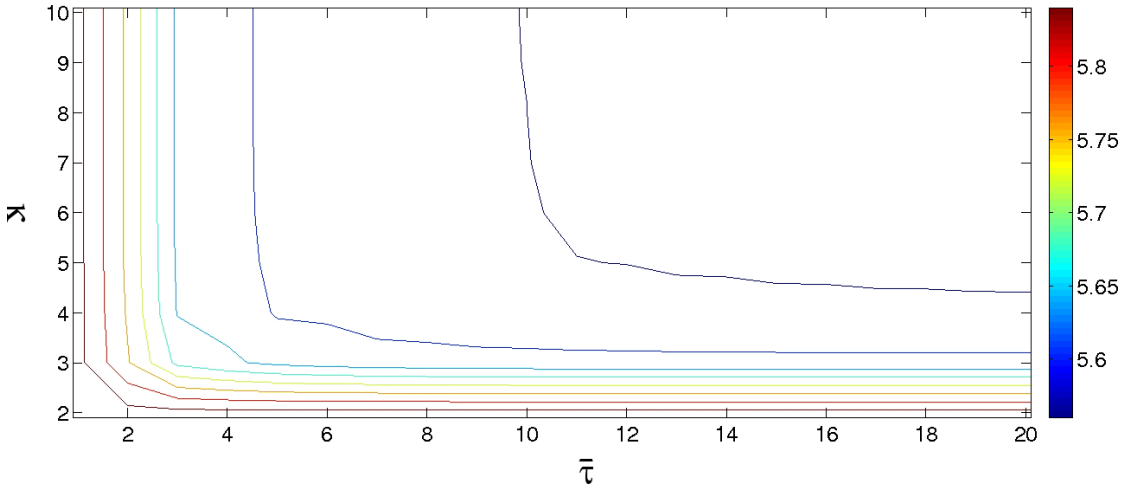


Figure 4: Average of the norm of agents' real time estimate errors at $k = 50$. These estimates are causal. The level sets show the trade-offs for different values of κ and $\bar{\tau}$.

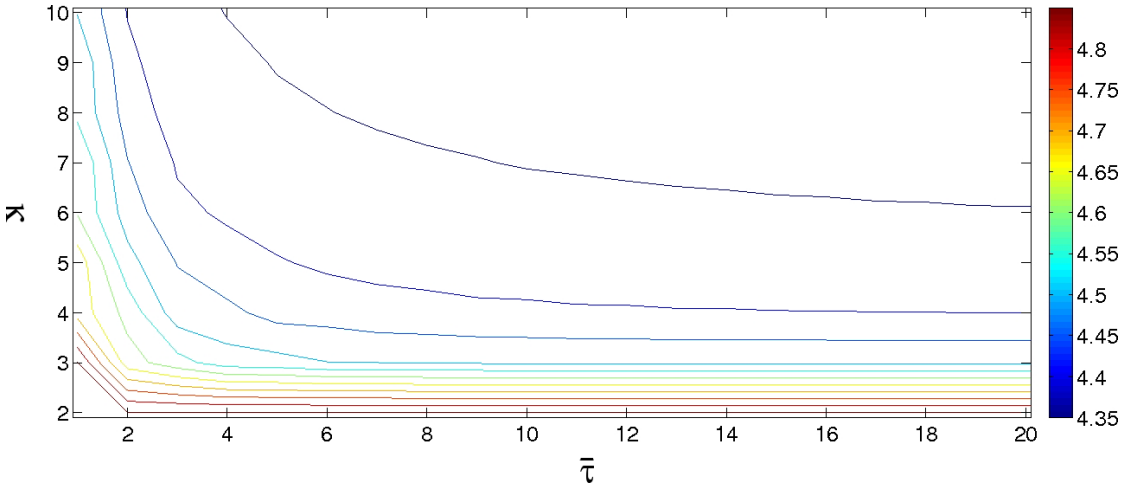


Figure 5: Average of the norm of agents' smoothed estimate errors at $k = 40$. These estimates are non-causal. The level sets show the trade-offs for different values of κ and $\bar{\tau}$.

5 Conclusion

In this paper we presented a distributed algorithm for mobile agents to estimate their positions by fusing their own displacement measurements with inter-agent relative position measurements. The algorithm is distributed in the sense each agent can estimate its own position by communicating with nearby agents. The algorithm computes an approximation of the centralized optimal (Kalman smoother) estimates. The problem of distributed Kalman smoothing for this application is reformulated as a problem of computing the BLUE estimates. The graph structure of the BLUE estimation problem, which makes it computable using iterative methods of solving linear equations, makes distributing the computations possible. With finite

memory and limited number of iterations before new measurements are obtained, the algorithm produces an approximation of the Kalman smoother estimates. As the memory of each agent and the number of iterations between each time step are increased, the approximation improves.

We extended this method to more general partitioning of the graph's node set. It was shown that by using a more general partitioning, the distributed estimation method can be extended to target tracking under a modest observability assumption. We provided a closed form bound for convergence of the algorithm on a lattice with no constraint on memory. Also, we illustrated the tradeoff between the number of iterations per time step and the amount of history retained during estimation. We showed that under very modest parameters, the algorithm vastly improves estimates when compared to dead reckoning. Furthermore, if the algorithm parameters are slightly improved performance of the algorithm nears that of Kalman Smoothing.

References

Alriksson, P. & Rantzer, A. (2006). Distributed Kalman filtering using weighted averaging. In 17th International Symposium on Mathematical Theory of Networks and Systems (MTNS).

Alriksson, P. & Rantzer, A. (2007). Experimental evaluation of a distributed Kalman filter algorithm. In 46th IEEE Conference on Decision and Control.

Barooah, P. & Hespanha, J. P. (2005). Distributed optimal estimation from relative measurements. In Proceedings of the 3rd International Conference on Intelligent Sensing and Information Processing (ICISIP) (pp. 226–231).

Barooah, P. & Hespanha, J. P. (2007). Estimation from relative measurements : Algorithms and scaling laws. IEEE Control Systems Magazine, 27(4), 57 – 74.

Barooah, P. & Hespanha, J. P. (2009). Error scaling laws for linear optimal estimation from relative measurements. 55(12), 5661–5673.

Barooah, P., Russell, W. J., & Hespanha, J. P. (2010). Approximate distributed Kalman filtering for cooperative multi-agent localization. In R. Rajaraman, T. Moscibroda, A. Dunkels, & A. Scaglione (Eds.), Distributed Computing in Sensor Systems (pp. 102–115). Springer.

Borenstein, J., Everett, H. R., Feng, L., & Wehe, D. (1997). Mobile robot positioning: Sensors and techniques. Journal of Robotic Systems, Special Issue on Mobile Robots, 14(4), 231–249.

Carli, R., Chiuso, A., Schenato, L., & Zampieri, S. (2007). Distributed Kalman filtering using consensus strategies. In 46th IEEE Conference on Decision and Control.

Feingold, D. G. & Varga, R. S. (1962). Block diagonally dominant matrices and generalizations of the gershgorin circle theorem. Pacific Journal of Mathematics, 12, 1241–1250.

Kurazume, R., Nagata, S., & Hirose, S. (1994). Cooperative positioning with multiple robots. In the IEEE International Conference in Robotics and Automation (pp. 1250–1257).

Makadia, A. & Daniilidis, K. (2005). Correspondenceless ego-motion estimation using an imu. In IEEE International Conference on Robotics and Automation (pp. 3534–3539).

Mendel, J. M. (1995). Lessons in Estimation Theory for Signal Processing, Communications and Control. Prentice Hall.

Mourikis, A. I. & Roumeliotis, S. I. (2006). Performance analysis of multirobot cooperative localization. IEEE Transactions on Robotics, 22(4), 666–681.

Mueller, S., Tsang, R. P., & Ghosal, D. (2004). Multipath routing in mobile ad hoc networks: Issues and challenges. In Performance Tools and Applications to Networked Systems, LNCS, volume 2965 (pp. 209–234). Springer Berlin/Heidelberg.

Nistér, D., Naroditsky, O., & Bergen, J. R. (2004). Visual odometry. In Conference on Computer Vision and Pattern Recognition (CVPR '04) (pp. 652–659).

Olfati-Saber, R. (2007). Distributed Kalman filtering for sensor networks. In 46th IEEE Conference on Decision and Control.

Olson, C. F., Matthies, L. H., Schoppers, M., & Maimone, M. W. (2003). Rover navigation using stereo ego-motion. Robotics and Autonomous Systems, 43(4), 215–229.

Oskiper, T., Zhu, Z., Samarasekera, S., & Kumar, R. (17-22 June 2007). Visual odometry system using multiple stereo cameras and inertial measurement unit. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR '07) (pp. 1–8).

Rekleitis, I. M., Dudek, G., & Milios, E. E. (2002). Multi-robot cooperative localization: a study of trade-offs between efficiency and accuracy. In the IEEE/RSJ International Conference on Intelligent Robots and System, volume 3 (pp. 2690–2695).

Roumeliotis, S. I. & Bekey, G. A. (2002). Distributed multirobot localization. IEEE Transactions on Robotics and Automation, (5), 781–795.

Spanos, D. P., Olfati-Saber, R., & Murray, R. M. (2005). Approximate distributed Kalman filtering in sensor networks with quantifiable performance. In 4th international symposium on Information processing in sensor networks (IPSN '05).

Trench, W. F. (1999). Solution 21-3.1 eigenvalues and eigenvectors of two symmetric matrices. Image: The Bulletin of the International Linear Algebra Society, 22, 22–23.

Zhang, P. & Martonosi, M. (2008). LOCALE: collaborative localization estimation for sparse mobile sensor networks. In International Conference on Information Processing in Sensor Networks (IPSN) (pp. 195–206).