Decentralised Solutions to the Cooperative Multi-Platform Navigation Problem

Mu Hua[a,b], Tim Bailey[a], Paul Thompson[a], Hugh Durrant-Whyte[a]

[a] *Australian Centre for Field Robotics, University of Sydney, NSW 2006, Australia*
[b] *Automatic Control Department, National University of Defense Technology, Hunan 410073, P.R.China*

Abstract

The problem of cooperative navigation for a team of platforms employing inter-platform observations is investigated. A decentralised solution in the framework of an information filter with delayed states is presented. In this structure, each platform first estimates its motion using only local sensor data, then shares its information across the network using an algorithm that employs a distributed Cholesky modification. The decentralised solution permits each platform to act in the same modular manner, providing robustness to individual platform failure. The solution yields linear minimum mean-square error estimation performance. As such the estimates generated are *optimal*; it generates exactly the same estimates as would a conventional extended Kalman filter, if given the same data. Efficient sparse implementation is accomplished without resorting to approximate methods. Simulation experiments employing a team of ten mobile platforms are described and used to evaluate the decentralised estimation performance. The robustness, flexibility and cost of the decentralised approach are analyzed and compared to an existing distributed solution.

Key words: Cooperative navigation, decentralised data fusion, delayed states, information-form Gaussian

1. Introduction

Navigation, the ability to determine position, velocity and attitude, is a fundamental capability of any mobile platform. Current navigation schema combine measurements from internal proprioceptive sensors, that monitor the motion of the platform, with landmark information collected by exteroceptive sensors that sense the external environment of the platform. However, where a team of platforms is required to accomplish a task, in applications such as search and rescue for example, the navigation solution for each platform is not independent if the same external landmarks are used by different platforms for self-localization, or if inter-platform observations are made. In these cases, navigation of the platform team needs to be treated as a whole, that is, *cooperative (coordinated,or collaborative) navigation* is required. Cooperative navigation of a platform team has some fundamental advantages over independent navigation of each platform [1]. For a homogeneous platform team, better estimates of external landmarks can be achieved by integrating measurements made by different platforms at many locations which can, in turn, improve the individual platforms navigation accuracy. For a heterogeneous team of platforms, a platform carrying low-precision navigation sensors can make use of high-precision navigation sensors hosted on other platforms to improve its navigation performance. Equally, a platform which can not accomplish a navigation

task by itself, due to limits in sensing or environments, can be 'navigation enabled' by cooperation with other platforms.

This paper introduces a decentralised solution to the general cooperative multi-platform navigation problem in the framework of the information filter [2] with delayed states. The delayed states are retained to capture historical dependencies or correlations between estimates, and to permit fusion of data involving multiple platforms. The decentralised structure proposed in this paper integrates distributed computing [3] and makes use of the distributed Cholesky modification for delayed states. In the Decentralised Data Fusion (DDF) framework, there is no central data fusion center and the platform team is able to reorganize itself when any single platform fails. The presented algorithm is an *optimal* estimator. It makes no approximations apart from the usual model linearisations and, once all of the available information has propagated through the system, will generate the same estimates as a centralised estimator, such as an EKF.

Simulation experiments employing a group of ten mobile platforms are described. These verify the proposed solution and evaluates its performance. The results demonstrate that the solution improves the navigation accuracy for platform team after integrating inter-platform observations. Comparing the decentralised solution with an existing distributed solution, the former has the advantage in terms of robustness while the latter has the advantage in terms of communication delay and computational cost.

The paper is organized as follows. Section 2 revisits previous work on cooperative localization. Section 3 formulates the cooperative navigation problem in the framework of the information filter with delayed states, and presents the operations for incremental modification of Cholesky factors. Section 4 presents the new decentralised algorithm for cooperative navigation. Section 5 provides results of the simulation experiments. Section 6 analyzes the performance of the decentralised approach comparing with a distributed solution. Conclusions and extensions are given in Section 7.

2. Related Work

Probabilistic approaches to the decentralised multi-platform navigation problem have been explored in a number of previous publications [4, 5, 6, 7]. While the methods described in [5, 6] are said to be 'distributed', we classify them here as decentralised. The difference between a distributed structure and a decentralised structure is that the former has a fusion center while the latter does not. In [5] a probabilistic algorithm for collaborative mobile robot localization is developed. The decentralised structure is realized by exploiting an approximation that removes interdependencies between robots. While the assumption makes the algorithm computationally tractable, it results in an inconsistent (over confident) solution. Evaluation of the algorithm is performed using only two robots. A method combining maximum likelihood estimation and distributed numerical optimization is presented in [6]. This approach provides estimates of relative, rather than global, platform localization. The paper argues (without proof) that the distributed optimization algorithm provides a solution equal to that obtained by centralised optimization. Experiments using a team of four mobile robots are presented. A decentralised Kalman filter framework, called 'collective localization' is proposed in [4]. A centralised Kalman filter is distributed as multiple Kalman filters each operating on one platform in the team. The key element of this method is to distribute the cross-correlation terms of the joint covariance matrix on different platforms. The experiments employ a group of three mobile robots. Although this method can be applied to a larger group in principle, the treatment of the covariance matrix (which is dense since the robots are correlated to each other) will not scale.

While the information filter and the Kalman filter are mathematically equivalent, they have different complexity characteristics for time and observation update steps. These two (equivalent) filters are thus suitable for different problems. The information filter (or extended information filter) is referred to in [4] and it's argued that 'For the case of distributed multirobot localization, the Kalman filter performs significantly better'. The same opinion is held in [7]. Two problems in the information filter applied to the cooperative navigation are mentioned to support these opinions [4]. The first problem is that large matrix inversion is required during each propagation step. The second problem is that the state recovery needed at every step involves the inverse of the whole information matrix.[1]

To address these two problems a decentralised structure in the framework of the information filter is proposed. The matrix inversion is avoided in the time propagation step by introducing delayed states into the information filter. With delayed states, the prediction becomes state augmentation which does not require the inversion of the information matrix. The state recovery is realized incrementally and only a submatrix instead of the complete information matrix is involved. The operation is implemented in constant time. Comparison with the treatment of the correlations in a dense covariance matrix shows much reduced computational complexity and better scalability.

The primary influence on this work is from recent research in Simultaneous Localization and Mapping (SLAM) [8, 9, 10, 11]. This literature demonstrates the strengths of the information filter in multi-sensor data fusion: State augmentation and observation updates are both constant time operations, fusion of information from different sensors is additive, and the key concept of delayed states can be introduced in a natural manner. Delayed states are used for landmark initialisation [12, 13, 14] in situations when data must be accumulated over a period of time before the landmark becomes fully observable. In [10], a full-smoothing approach is adopted retaining all the localization states of the platform over time. The introduction of delayed states in the work described in this paper avoids the need to compute the inverse information matrix in the time-update (prediction) step of the navigation filter, manages historical dependencies and permits fusion of inter-platform observations. In this form, Cholesky modification can be used as an efficient way to recover state mean and covariance information as proposed in [10] and implemented in an application of large scale view-based SLAM [15].

An approach to solving the cooperative navigation problem using the information filter framework with delayed states is described in [16]. This demonstrates the advantages of the additivity property of the observation update step of the information filter and how delayed states can be used to fuse inter-platform observations. The structure described is ego-centric in which every platform acts as a fusion centre, duplicating data fusion rather than being fully decentralised.

Another significant influence on the work described in this paper is distributed computing methods [3] and parallel computing methods [17]. It is noted, however, that distributed computing can only contribute general theoretical foundations and that further progress needs to be made concerning the application of such methods to a dynamic problem [18]. The introduction of a distributed Cholesky modification provides a solution to the multi-platform navigation problem in a decentralised structure.

[1] In [4] a third problem cited is that the covariance matrix may lose rank in some cases, e.g., when two platforms stand still and keep measuring range between themselves. In this case, however, both the information and Kalman filters will fail.

3. Problem Formulation and Representation

In this section the cooperative multi-platform navigation problem is formulated, the properties of information filter with delayed states are introduced and the operations for incremental Cholesky modification are provided.

3.1. Problem Statement

The cooperative multi-platform navigation may be stated as follows:

1. A team of N mobile platforms move in an environment. Each platform can move independently according to a local dynamic model.
2. Each platform is equipped with dead-reckoning sensors to measure self-motion; odometry or inertial sensors for example.
3. Some platforms may also be equipped with external sensors, such as GPS, to correct dead-reckoning estimates. These sensors provide observations involving only one platform.
4. Each platform also carries sensors that provide inter-platform measurements such as range or bearing to each other or to a common landmark. This type of of sensor provides observations involving two or more platforms in the team.
5. All platforms are equipped with communication devices allowing team members to exchange information with each other.

The cooperative navigation task requires each platform to estimate its own state (position, orientation, velocity etc) making use of both its own observations and those observations made by and of other platforms. If required, a platform can maintain estimates of the states of the other members in the team. The cooperative navigation problem is complex as the estimated states of the team members are correlated through measurements of common states. Conversely, these correlations make cooperative navigation valuable in improving navigation accuracy for all team members and enabling platforms without sufficient sensing to navigate using information from other team members.

In this work, each platform is equipped with odometry and a ranging sensor. Some have independent GPS-like position sensors, other do not. The platforms are identified by capitals $A, B, C, \cdots$.

3.2. Information Augmentation and Fusion

Appendix A defines the equivalent covariance-form $\mathcal{N}_S(\mathbf{x}; \hat{\mathbf{x}}, \mathbf{P})$ (Equation A-1) and information-form $\mathcal{N}_I(\mathbf{x}; \hat{\mathbf{y}}, \mathbf{Y})$ (Equation A-2) parametric forms for Gaussians. The information matrix $\mathbf{Y}$ and covariance matrix $\mathbf{P}$ are related as $\mathbf{Y} = \mathbf{P}^{-1}$ and the information vector $\hat{\mathbf{y}}$ and the state mean $\hat{\mathbf{x}}$ are related as $\hat{\mathbf{y}} = \mathbf{P}^{-1}\hat{\mathbf{x}}$. Appendix A also shows that information fusion is simply addition in the information form and marginalisation (prediction) requires inversion of the information matrix. With these results, we describe the process of augmentation and observation fusion in information form for the navigation problem.

The platform state at time k is denoted $\mathbf{x}_k$. The state of the platform is assumed to evolve according to the motion equation

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k) + \mathbf{q}, \qquad \mathbf{q} \sim \mathcal{N}_S(\mathbf{q}; \mathbf{0}, \mathbf{Q}) = \mathcal{N}_I(\mathbf{q}; \mathbf{0}, \mathbf{Q}^{-1}).$$

State Augmentation: A joint state-vector $\mathbf{x} = [\mathbf{x}_{k-1}^T, {\mathbf{x}_k}^T]^T$ comprising the platform states at time $k-1$ and k. This joint state-vector is assumed Gaussian and parameterised in information form as $\mathcal{N}_I(\mathbf{x}; \hat{\mathbf{y}}, \mathbf{Y})$ with

$$\hat{\mathbf{y}} = \begin{bmatrix} \hat{\mathbf{y}}_{k-1} \\ \hat{\mathbf{y}}_k \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} \mathbf{Y}_{k-1,k-1} & \mathbf{Y}_{k-1,k} \\ \mathbf{Y}_{k-1,k}^T & \mathbf{Y}_{k,k} \end{bmatrix}. \tag{1}$$

An augmented state vector $\mathbf{x}_a$ is now constructed by joining the state $\mathbf{x}_{k+1}$ to the joint-state $\mathbf{x}$:

$$\mathbf{x}_a^T = \left[\begin{array}{cc} \mathbf{x}^T & \mathbf{x}_{k+1}^T \end{array} \right] = \left[\begin{array}{ccc} \mathbf{x}_{k-1}^T & \mathbf{x}_k^T & \mathbf{x}_{k+1}^T \end{array} \right] = \left[\begin{array}{ccc} \mathbf{x}_{k-1}^T & \mathbf{x}_k^T & (\mathbf{f}(\mathbf{x}_k) + \mathbf{q})^T \end{array} \right] \tag{2}$$

From Equations (A-20, A-21), the information parameters for the augmented state vector are

$$\hat{\mathbf{y}}_a = \left[\begin{array}{c} \hat{\mathbf{y}}_{k-1} \\ \hat{\mathbf{y}}_k - \mathbf{F}_k^T \mathbf{Q}^{-1} [\mathbf{f}(\hat{\mathbf{x}}_k) - \mathbf{F}_k \hat{\mathbf{x}}_k] \\ \mathbf{Q}^{-1} [\mathbf{f}(\hat{\mathbf{x}}_k) - \mathbf{F}_k \hat{\mathbf{x}}_k] \end{array} \right], \tag{3}$$

$$\mathbf{Y}_a = \left[\begin{array}{ccc} \mathbf{Y}_{k-1,k-1} & \mathbf{Y}_{k-1,k} & \mathbf{0} \\ \mathbf{Y}_{k-1,k}^T & \mathbf{Y}_{k,k} + {\mathbf{F}_k}^T \mathbf{Q}^{-1} \mathbf{F}_k & -\mathbf{F}_k^T \mathbf{Q}^{-1} \\ \mathbf{0} & -\mathbf{Q}^{-1} \mathbf{F}_k & \mathbf{Q}^{-1} \end{array} \right], \tag{4}$$

where the non-linear motion model has been appropriately linearised with $\mathbf{F}_k = \frac{\partial \mathbf{f}}{\partial \mathbf{x}_k}\Big|_{\hat{\mathbf{x}}_k}$.

There are two important points to note about these expressions for the augmented information parameters. First, there is no requirement to invert an information matrix. Second, the cross-information between $\mathbf{x}_{k+1}$ and $\mathbf{x}_{k-1}$ is zero. Indeed, the cross-information between $\mathbf{x}_{k+1}$ and all states $\mathbf{x}_j$, where $j \neq k, k+2$, is zero and thus the joint information matrix is block-diagonal and sparse in general. These two points explain why the augmented information form is potentially computationally efficient.

The sub-states $\mathbf{x}_{k-1}$ can be removed from the joint estimation $\mathbf{x} = [\mathbf{x}_{k-1}^T, {\mathbf{x}_k}^T]^T$ by *marginalisation* using the following information-form expressions,

$$\tilde{\mathbf{y}}_k = \hat{\mathbf{y}}_k - \mathbf{Y}_{k-1,k}^T \mathbf{Y}_{k-1,k-1}^{-1} \hat{\mathbf{y}}_{k-1}, \tag{5}$$

$$\tilde{\mathbf{Y}}_{k,k} = \mathbf{Y}_{k,k} - \mathbf{Y}_{k-1,k}^T \mathbf{Y}_{k-1,k-1}^{-1} \mathbf{Y}_{k-1,k}. \tag{6}$$

Note, marginalisation will result in the information matrix losing its sparseness property and therefore is not undertaken unless required.

Observation Updates for Individual Platforms: At time k the platform is assumed to make measurements according to the measurement equation

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{r}, \qquad \mathbf{r} \sim \mathcal{N}_S(\mathbf{r}; \mathbf{0}, \mathbf{R}) = \mathcal{N}_I(\mathbf{r}; \mathbf{0}, \mathbf{R}^{-1}).$$

In information form, fusion (a product of probability densities) is simply addition, so the updated information vector $\hat{\mathbf{y}}_k^+$ and matrix $\mathbf{Y}_k^+$ are given by

$$\hat{\mathbf{y}}_k^+ = \left[\begin{array}{c} \hat{\mathbf{y}}_{k-1} \\ \hat{\mathbf{y}}_k + \mathbf{i}_k \end{array} \right], \tag{7}$$

$$\mathbf{Y}_k^+ = \left[\begin{array}{cc} \mathbf{Y}_{k-1,k-1} & \mathbf{Y}_{k-1,k} \\ {\mathbf{Y}_{k-1,k}}^T & \mathbf{Y}_{k,k} + \mathbf{I}_k \end{array} \right], \tag{8}$$

where the observation information vector and matrix are given by

$$\mathbf{i}_k = \mathbf{H}^T \mathbf{R}^{-1} \left(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_k) + \mathbf{H} \hat{\mathbf{x}}_k \right), \qquad \mathbf{I}_k = \mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}$$

and where $\mathbf{H} = \frac{\partial \mathbf{h}}{\partial \mathbf{x}_k}\Big|_{\hat{\mathbf{x}}_k}$.

The important point to note here is that fusion in the augmented information state affects only the sub-vector associated with the current time. This means that the fusion

operation is sparse and computationally trivial. This should be compared to the equivalent operation for a covariance-based parameterisation (a Kalman filter) which would require the calculation of a gain matrix and which results in a non-sparse update.

Observation Updates for Several Platforms: Consider now a state vector $\mathbf{x}$ comprising sub-state vectors of three platforms A, B and C, $[\mathbf{x}_A^T, \mathbf{x}_B^T, \mathbf{x}_C^T]^T$ (the time subscript k is omitted here to aid understanding of the essential update process). Consider an observation taken between platforms A and B; a relative range for example or an observation of a common landmark. The observation model is given by,

$$\mathbf{z} = \mathbf{h}\left(\mathbf{x}_A, \mathbf{x}_B\right) + \mathbf{r} \qquad \mathbf{r} \sim \mathcal{N}_S(\mathbf{r}; \mathbf{0}, \mathbf{R}) = \mathcal{N}_I(\mathbf{r}; \mathbf{0}, \mathbf{R}^{-1}). \tag{9}$$

The updated information vector and matrix are given by

$$\hat{\mathbf{y}}_k^+ = \begin{bmatrix} \hat{\mathbf{y}}_A + \mathbf{i}_A \\ \hat{\mathbf{y}}_B + \mathbf{i}_B \\ \hat{\mathbf{y}}_C \end{bmatrix}, \tag{10}$$

$$\mathbf{Y}_k^+ = \begin{bmatrix} \mathbf{Y}_{AA} + \mathbf{I}_{AA} & \mathbf{Y}_{AB} + \mathbf{I}_{AB} & \mathbf{Y}_{AC} \\ \mathbf{Y}_{AB}^T + \mathbf{I}_{AB}^T & \mathbf{Y}_{BB} + \mathbf{I}_{BB} & \mathbf{Y}_{BC} \\ \mathbf{Y}_{AC}^T & \mathbf{Y}_{BC}^T & \mathbf{Y}_{CC} \end{bmatrix}, \tag{11}$$

where the observation information vector and matrix are given by

$$\mathbf{i}_i = \mathbf{H}_i^T \mathbf{R}^{-1} \left(\mathbf{z} - \mathbf{h}(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j) + \mathbf{H}_i \hat{\mathbf{x}}_i + \mathbf{H}_j \hat{\mathbf{x}}_j\right), \qquad \mathbf{I}_{ij} = \mathbf{H}_i^T \mathbf{R}^{-1} \mathbf{H}_j$$

and where $\mathbf{H}_i = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}_i}\right|_{(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j)}$.

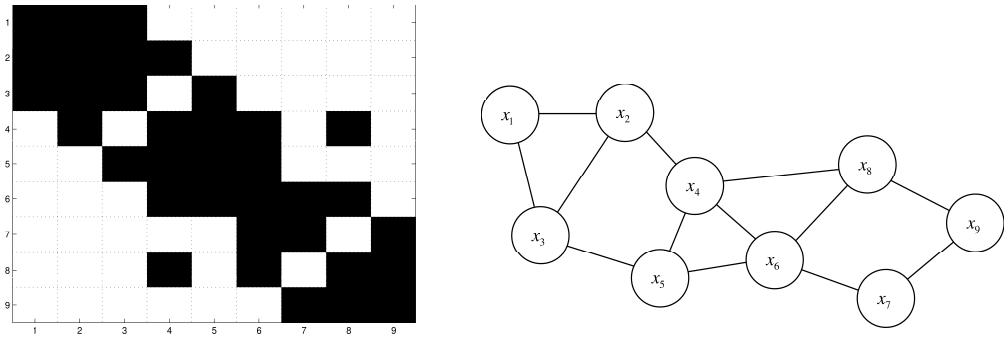
As before, the information update only affects those components of the information vector and information matrix directly related to the observation; regardless of how the states are correlated. This should be contrasted to the covariance-form update in which all observable states would be updated by an observation through the estimated cross correlation. The advantage of the information parameterisation is again clear. The update required to platforms from a common observation is local only to those platforms and need not be propagated to other platforms in the team at *observation time*. This ensures that both computation and communication are minimised.

3.3. Graphical Model Representation

To assist in the exposition in Section 4, we exploit the equivalence of the information form Gaussian and the graphical model for Markov random fields [19, 16]. The vertices of a Markov random field represent the diagonal entries of the information matrix and the edges (or links) represent the off-diagonal entries, as shown in Figure 1. If there is no link between two vertices, the associated off-diagonal terms are zeros. A vertex may also represent a *group* of states, in which case it corresponds to a block-diagonal component of the information matrix, and its links correspond to a block of off-diagonal terms. The vertices also represent the terms of the information vector. Thus, changing the value of a vertex is equivalent to changing the value of a (block-) diagonal entry of the information matrix, and the associated terms of the information vector, and changing the value of a link is equivalent to changing the value of the associated off-diagonal entries of the information matrix.

3.4. Estimation with Cholesky Factors

The algorithms for decentralised estimation and state moment recovery presented in Section 4 are implemented using a Cholesky factored form of the information matrix. The essential operations for manipulating the joint state information and recovering partial estimates of the mean and covariance are described below.



(a) Information matrix (b) Graphical model

Figure 1: An example information matrix (a), with non-zero entries marked in black, and its corresponding graphical model (b). The vertices in (b) are also representative of the information vector.

3.4.1. Block-based Cholesky Modification

Consider a system $\{\hat{\mathbf{y}}, \mathbf{Y}\}$ which is partitioned in 4×4 blocks. The Cholesky factor of $\mathbf{Y}$ has the following form,

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_{11} & & & \\ \mathbf{L}_{21} & \mathbf{L}_{22} & & \\ \mathbf{L}_{31} & \mathbf{L}_{32} & \mathbf{L}_{33} & \\ \mathbf{L}_{41} & \mathbf{L}_{42} & \mathbf{L}_{43} & \mathbf{L}_{44} \end{bmatrix}$$

where each $\mathbf{L}_{ii} (i = 1, 2, 3, 4)$ is itself a lower triangular matrix. The following notations are used. $\mathbf{L}_{i*}$ is the i^{th} row block; $\mathbf{L}_{*j}$ is the j^{th} column block; the size of $\mathbf{L}_{ij}$ is $r_i \times c_j$, where r_i is the number of rows and c_j is the number of columns. Similar notations are used for $\mathbf{Y}$.

The block-based up-looking Cholesky factorization for $\mathbf{Y}_{4*}$ is accomplished in two steps [17]. The off-diagonal blocks are the solution of a triangular system

$$\begin{bmatrix} \mathbf{L}_{11} & & \\ \mathbf{L}_{21} & \mathbf{L}_{22} & \\ \mathbf{L}_{31} & \mathbf{L}_{32} & \mathbf{L}_{33} \end{bmatrix} \begin{bmatrix} \mathbf{L}_{41}^T \\ \mathbf{L}_{42}^T \\ \mathbf{L}_{43}^T \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{41}^T \\ \mathbf{Y}_{42}^T \\ \mathbf{Y}_{43}^T \end{bmatrix}. \tag{12}$$

The diagonal block is computed from a Cholesky factorization

$$\mathbf{L}_{44}\mathbf{L}_{44}^T = \mathbf{Y}_{44} - \begin{bmatrix} \mathbf{L}_{41} & \mathbf{L}_{42} & \mathbf{L}_{43} \end{bmatrix} \begin{bmatrix} \mathbf{L}_{41} & \mathbf{L}_{42} & \mathbf{L}_{43} \end{bmatrix}^T. \tag{13}$$

If the modification of $\mathbf{L}$ starts from $\mathbf{L}_{*3}$, then only rows 3 and 4 are affected, where the modified part for $\mathbf{L}_{3*}$ is calculated from a Cholesky factorization

$$\mathbf{L}_{33}\mathbf{L}_{33}^T = \mathbf{Y}_{33} - \begin{bmatrix} \mathbf{L}_{31} & \mathbf{L}_{32} \end{bmatrix} \begin{bmatrix} \mathbf{L}_{31} & \mathbf{L}_{32} \end{bmatrix}^T, \tag{14}$$

and the modification for $\mathbf{L}_{4*}$ is

$$\begin{aligned} \mathbf{L}_{33}\mathbf{L}_{43}^T &= \mathbf{Y}_{43}^T - \begin{bmatrix} \mathbf{L}_{31} & \mathbf{L}_{32} \end{bmatrix} \begin{bmatrix} \mathbf{L}_{41} & \mathbf{L}_{42} \end{bmatrix}^T \\ \mathbf{L}_{44}\mathbf{L}_{44}^T &= \mathbf{Y}_{44} - \begin{bmatrix} \mathbf{L}_{41} & \mathbf{L}_{42} & \mathbf{L}_{43} \end{bmatrix} \begin{bmatrix} \mathbf{L}_{41} & \mathbf{L}_{42} & \mathbf{L}_{43} \end{bmatrix}^T. \end{aligned} \tag{15}$$

Property 1. From (14, 15) it is apparent that if the changed part of $\mathbf{L}$ starts from $\mathbf{L}_{*j}$, then rows $\mathbf{L}_{k*}$, where $k < j$, do not appear in the modification calculations. □

It can be derived from (12) that the leftmost non-zero element of $\mathbf{L}_{i*}$ shows up at the same column of the leftmost non-zero element of $\mathbf{Y}_{i*}$. If $\mathbf{Y}_{41} = 0$, then $\mathbf{L}_{41} = 0$, and the remainder of $\mathbf{L}_{4*}$ is calculated as

$$\begin{bmatrix} \mathbf{L}_{22} & \\ \mathbf{L}_{32} & \mathbf{L}_{33} \end{bmatrix} \begin{bmatrix} \mathbf{L}_{42}^T \\ \mathbf{L}_{43}^T \end{bmatrix} = \begin{bmatrix} \mathbf{Y}_{42}^T \\ \mathbf{Y}_{43}^T \end{bmatrix}$$

$$\mathbf{L}_{44}\mathbf{L}_{44}^T = \mathbf{Y}_{44} - \begin{bmatrix} \mathbf{L}_{42} & \mathbf{L}_{43} \end{bmatrix} \begin{bmatrix} \mathbf{L}_{42} & \mathbf{L}_{43} \end{bmatrix}^T. \tag{16}$$

The computational cost for this calculation is $O(r_4(r_2^2 + r_3^2)) + O(r_3 r_4 c_2) + O(r_4^2(c_2 + c_3)) + O(c_4^3)$.

Property 2. Equation (16) shows that if the leftmost nonzero element of $\mathbf{Y}_{i*}$ appears at position $\mathbf{Y}_{ij}$, where $1 < j < i$, then $\mathbf{L}_{*k}$, for $k < j$, are not needed in the calculation of $\mathbf{L}_{i*}$. □

Furthermore, when $\mathbf{L}_{41} = 0$ and $\mathbf{L}_{42}$ is unchanged, the modification of $\mathbf{L}_{4*}$ simplifies to

$$\mathbf{L}_{33}\mathbf{L}_{43}^T = \mathbf{Y}_{43}^T - \mathbf{L}_{32}\mathbf{L}_{42}^T$$

$$\mathbf{L}_{44}\mathbf{L}_{44}^T = \mathbf{Y}_{44} - \begin{bmatrix} \mathbf{L}_{42} & \mathbf{L}_{43} \end{bmatrix} \begin{bmatrix} \mathbf{L}_{42} & \mathbf{L}_{43} \end{bmatrix}^T. \tag{17}$$

The computational cost in this case is $O(r_3^2 r_4) + O(r_3 r_4 c_2) + O(r_4^2(c_2 + c_3)) + O(c_4^3)$.

3.4.2. Partial State Mean Recovery Algorithm

Given $\mathbf{L}$, the complete state mean vector $\hat{\mathbf{x}}$ can be recovered in two steps: forward substitution and backward substitution.

$$\mathbf{L}\hat{\mathbf{f}} = \hat{\mathbf{y}}, \tag{18}$$

$$\mathbf{L}^T\hat{\mathbf{x}} = \hat{\mathbf{f}}, \tag{19}$$

where $\hat{\mathbf{f}}$ is the solution of the forward substitution.

Assume the system $\{\hat{\mathbf{y}}, \mathbf{Y}\}$ is partitioned into three blocks

$$\begin{bmatrix} \mathbf{L}_{11} & & \\ \mathbf{L}_{21} & \mathbf{L}_{22} & \\ \mathbf{L}_{31} & \mathbf{L}_{32} & \mathbf{L}_{33} \end{bmatrix} \begin{bmatrix} \hat{\mathbf{f}}_1 \\ \hat{\mathbf{f}}_2 \\ \hat{\mathbf{f}}_3 \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{y}}_1 \\ \hat{\mathbf{y}}_2 \\ \hat{\mathbf{y}}_3 \end{bmatrix}. \tag{20}$$

If a modification is performed on the last block $\{\hat{\mathbf{y}}_3^+, \mathbf{L}_{33}^+\}$ only, the change in forward substitution and subsequent recovery of $\hat{\mathbf{x}}_3^+$ is computed as

$$\mathbf{L}_{33}^+\hat{\mathbf{f}}_3^+ = \hat{\mathbf{y}}_3^+ - \mathbf{L}_{31}\hat{\mathbf{f}}_1 - \mathbf{L}_{32}\hat{\mathbf{f}}_2, \tag{21}$$

$$\mathbf{L}_{33}^{+T}\hat{\mathbf{x}}_3^+ = \hat{\mathbf{f}}_3^+. \tag{22}$$

Considering the sparseness of $\mathbf{L}$, say $\mathbf{L}_{31} = \mathbf{0}$, (21) simplifies to

$$\mathbf{L}_{33}^+\hat{\mathbf{f}}_3^+ = \hat{\mathbf{y}}_3^+ - \mathbf{L}_{32}\hat{\mathbf{f}}_2. \tag{23}$$

In this case, only the subvector $\hat{\mathbf{f}}_2$ is needed, where the length of $\hat{\mathbf{f}}_2$ is c_2, and so the number of elements in $\mathbf{L}$ involved in partial mean recovery is constant; the operation can be implemented in constant time. The computational cost involved in (23) is $O(r_3 c_2) + O(c_3^2)$ and in (22) is $O(c_3^2)$.

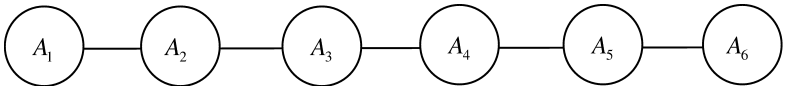


Figure 2: Platform A augments a Markov chain of its state history every time-step, fusing odometry and GPS. Each vertex A_k of the chain represents the platform's instantaneous state $\mathbf{x}_k$ at time-step k.

3.4.3. Partial State Covariance Recovery Algorithm

While the information matrix for the cooperative navigation system is sparse, the corresponding covariance matrix $\mathbf{P}$ is dense. To recover the whole $\mathbf{P}$ is computationally expensive and not necessary. What are needed in practical applications are the diagonal or block-diagonal elements. Computing the elements of $\mathbf{P}$ (whose size is $n \times n$) only at the locations coincident with the non-zeros of $\mathbf{L}$ constitutes a 'sparse inverse', which can be calculated as follows [20]

$$\begin{aligned} p_{jj} &= \left(l_{jj}^{-1} - \sum_{k=j+1, l_{kj}\neq 0}^{n} p_{kj} l_{kj} \right) l_{jj}^{-1}, \\ p_{ij} &= -\left(\sum_{k=j+1, l_{kj}\neq 0}^{i} p_{ik} l_{kj} + \sum_{k=i+1, l_{kj}\neq 0}^{n} p_{ki} l_{kj} \right) l_{jj}^{-1} \\ & \qquad (j = 1, 2, \cdots, n, \quad j < i, \quad l_{ij} \neq 0). \end{aligned} \tag{24}$$

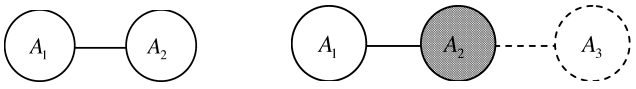
Starting from the bottom-right element of $\mathbf{L}$, individual elements of the sparse inverse can be calculated recursively from right to left, from bottom to top. To recover elements in the submatrix $\mathbf{P}_{m:n,m:n}$, where $(1 \leqslant m < n)$, only the submatrix $\mathbf{L}_{m:n,m:n}$ is involved. If $\mathbf{L}_{m:n,m:n}$ has k nonzeros, the cost of recovering the sparse inverse is $O(k^3)$, and so, for constant k, partial covariance recovery is a constant time operation.

4. A Decentralised Solution to Cooperative Navigation

This section presents a particular solution to the cooperative navigation problem, and is the key contribution of this paper. There are essentially four parts to this algorithm. The first is local information assimilation, where each platform computes a Markov chain estimate of its own state history in information form. The second is maintaining the joint state of all platforms in Cholesky form; the full state history is never located on any one platform, but is distributed across the network. Third, any given platform can compute optimal mean and covariance estimates of recent states. And fourth, old (obsolete) states are deleted to limit storage and communication costs.

4.1. Local State Estimation

Each platform maintains an estimate of its own state given all available *local* information. Motion information from odometry and position information from onboard GPS are fused to form a Markov chain of the platform's time history $[\mathbf{x}_{A1}^T, \ldots, \mathbf{x}_{Ak}^T]^T$, as shown in Figure 2, where each $\mathbf{x}_{Ak}$ denotes the instantaneous platform state (position, velocity, etc.) at time k. This state for platform A is represented in information form as $\{\hat{\mathbf{y}}_A, \mathbf{Y}_A\}$, where $\mathbf{Y}_A$ is a sparse band-diagonal matrix.



(a) Before (b) After

Figure 3: The graphical model for platform A before and after the augmentation by motion information from time 2 to time 3. The augmentation operation alters the value of vertex A_2 (shaded), and adds a new link and vertex (dashed).

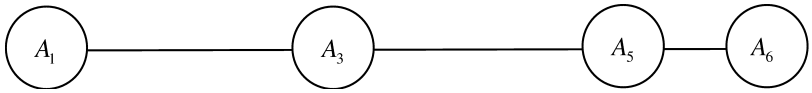


Figure 4: There were no range measurements taken by or of platform A at timesteps 2 and 4. These obsolete states are removed by marginalisation.

4.1.1. Position and Motion Assimilation

Fusion of GPS-like position information is trivial, since this information affects only the vertex (i.e., the block-diagonal entries of the information matrix and associated terms of the information vector) of the instant when the measurement was taken. Thus, if platform A obtains a GPS measurement at time 2, it updates only vertex A_2 according to observation model $\mathbf{z} = \mathbf{h}_{gps}(\mathbf{x}_{A2})$ and the update equations (7, 8).

The assimilation of motion information (such as odometry) involves the Markov model $\mathbf{x}_{Ak+1} = \mathbf{f}(\mathbf{x}_{Ak})$ and the augmentation operations of (3, 4). Augmentation extends the state-space by creating a new vertex and linking it to the previous end-state, as shown in Figure 3. The operation also changes the value of the existing vertex to which the new link connects. Both the new and old values of the existing vertex are stored, as both are required to facilitate calculations with and without augmentation information, as discussed in Section 4.3.2 (see especially Figure 12).

4.1.2. Removal of Irrelevant States

For the task of cooperative navigation, the key purpose of retaining historical states is to permit fusion of range measurements between platforms. A platform must retain a position state for every moment when either it observes, or is observed by, another platform. Conversely, any platform states that do *not* have an associated range measurement are obsolete and can be removed by the marginalisation equations (5, 6). The remaining states still form a Markov chain, as shown in Figure 4.

4.1.3. Range Fusion

When platform A observes the range to platform B at time k, it communicates to platform B the range measurement $\mathbf{z}_k$ and an estimate of its instantaneous position $\hat{\mathbf{x}}_{Ak}$.

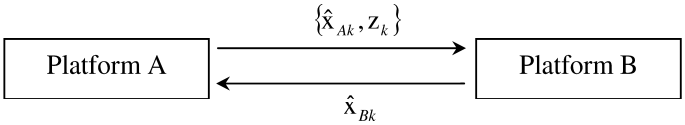


Figure 5: When platform A observes the range to platform B, there is an exchange between the two so that each knows $\{\hat{\mathbf{x}}_{Ak}, \hat{\mathbf{x}}_{Bk}, \mathbf{z}_k\}$.

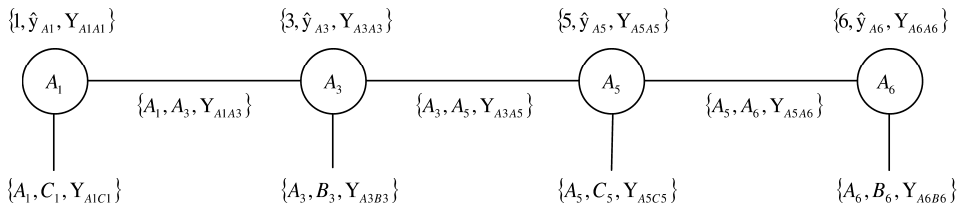


Figure 6: The content of the vertices and edges of a local state graph, including unconnected links to other platforms.

On receiving this information, platform B replies with an estimate of its instantaneous position $\hat{\mathbf{x}}_{Bk}$ at that time-step, as shown in Figure 5. This information allows both platforms to compute a range observation update with model $\mathbf{z} = \mathbf{h}_r(\mathbf{x}_{Ak}, \mathbf{x}_{Bk})$ as per (10, 11).

However, range fusion is not performed as a conventional information-form update, since neither platform retains information of the other platform. Each platform is effectively performing half of the update, and does not compute or store the information for the other state. Thus, platform A will store just

$$\begin{aligned}\hat{\mathbf{y}}^+_{Ak} &= \hat{\mathbf{y}}_{Ak} + \mathbf{H}_A^T\mathbf{R}^{-1}(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{Ak}, \hat{\mathbf{x}}_{Bk}) + \mathbf{H}_A\hat{\mathbf{x}}_{Ak} + \mathbf{H}_B\hat{\mathbf{x}}_{Bk}),\\ \mathbf{Y}^+_{Ak} &= \left[\mathbf{Y}_{AA} + \mathbf{H}_A^T\mathbf{R}^{-1}\mathbf{H}_A,\ \mathbf{Y}_{AB} + \mathbf{H}_A^T\mathbf{R}^{-1}\mathbf{H}_B\right],\end{aligned}$$

while platform B stores the other half of the update

$$\begin{aligned}\hat{\mathbf{y}}^+_{Bk} &= \hat{\mathbf{y}}_{Bk} + \mathbf{H}_B^T\mathbf{R}^{-1}(\mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{Ak}, \hat{\mathbf{x}}_{Bk}) + \mathbf{H}_A\hat{\mathbf{x}}_{Ak} + \mathbf{H}_B\hat{\mathbf{x}}_{Bk}),\\ \mathbf{Y}^+_{Bk} &= \left[\mathbf{Y}_{AB}^T + \mathbf{H}_B^T\mathbf{R}^{-1}\mathbf{H}_A,\ \mathbf{Y}_{BB} + \mathbf{H}_B^T\mathbf{R}^{-1}\mathbf{H}_B\right].\end{aligned}$$

These 'half estimates' linking two platforms are not trivial to realise with the usual matrix-like data structures. The local estimation process is easier to think about, and to implement, in terms of a graph data-structure that manages both the information-form data and bookkeeping for connections within and between platforms. For the example in Figure 6 each vertex k holds $\{k, \hat{\mathbf{y}}_{Ak}, \mathbf{Y}_{AkAk}\}$ (i.e., records for time, information vector, and information matrix block-diagonal), and each edge kj holds $\{A_k, X_j, \mathbf{Y}_{AkXj}\}$ (i.e., platform IDs, and information matrix off-diagonal terms). With this data-structure, the information is ready to slot into the correct location of the full (all platforms) joint state vector, as discussed below.

4.2. Determining Joint State Order

When the joint state of all platforms is depicted as a graph (see Figure 7), the links between vertices of the same platform are formed from motion information, and the links between different platforms are formed by range measurements. This graphical representation is sufficient for the information form estimate $\{\hat{\mathbf{y}}, \mathbf{Y}\}$, since the vertices and edges can directly represent the values of the information vector and matrix, and these values are not affected by the *order* of the elements in the state vector. However, for the Cholesky factor $\mathbf{L}$ of the information matrix, the order of the state vector *does* affect the value of its elements, so a graphical representation is not sufficient. We need a *specific* state order.

For decentralised estimation all platforms must agree on a single common state ordering. This order may be arbitrary, in principle, but, to simplify consensus between

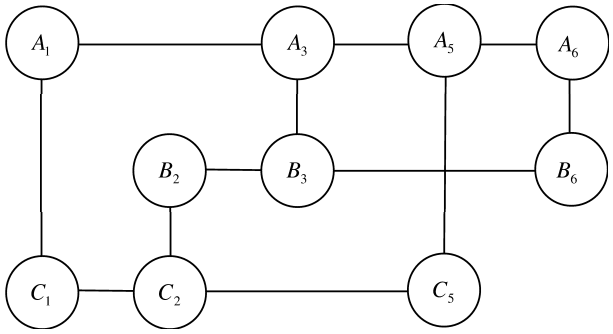


Figure 7: Joint historical state for three platforms A, B and C up to time-step 6. The horizontal links are formed by motion information and the vertical links by range information.

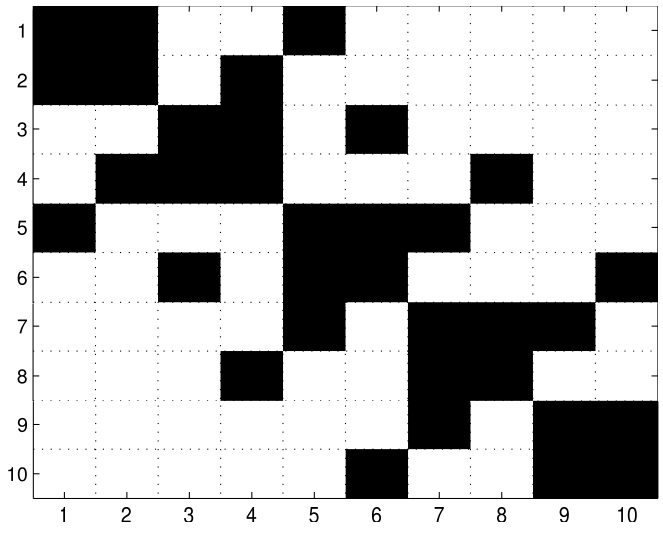


Figure 8: The information matrix for the graph in Figure 7 with the state ordering defined in (25).

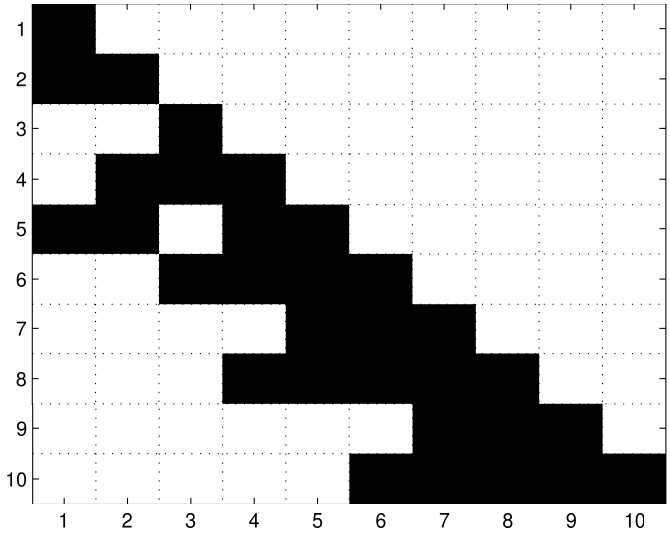


Figure 9: The Cholesky factor of the matrix in Figure 8. The left-most non-zero term of each row in $\mathbf{Y}$ determines the maximum possible fill-in for that row in $\mathbf{L}$. This gives an upper bound to the cost of Cholesky modification, as described in Section 4.3.

platforms, a predetermined ordering is used here: states are arranged in time-step order and states with the same time-stamp are arranged in order of platform ID. For the example in Figure 7, the state vector is ordered as

$$\{A_1, C_1, B_2, C_2, A_3, B_3, A_5, C_5, A_6, B_6\}\,. \tag{25}$$

Given this ordering, the joint information matrix looks like Figure 8 and its Cholesky factor matrix is as in Figure 9. Notice the fill-in pattern in $\mathbf{L}$; new non-zero entries only appear to the right of the left-most non-zero term in the corresponding row of $\mathbf{Y}$, as per (16) in Section 3.4.1.

State order determination is accomplished as follows. When a platform makes a range measurement at time k, it knows that the instantaneous states of both platforms must appear in the joint state vector. Therefore, it broadcasts a message to *all other platforms* with the platform identifiers and measurement time-stamp (e.g., $\{A, B, k\}$). This information is sufficient for all platforms to construct (25) based on the ordering rule.

Definition. The state order sequence, as e.g. in (25), is called the '***augmentation queue***' because it determines the sequence in which platforms incrementally augment the joint state vector. □

4.3. Distributing the Joint State

The essential process of building and distributing the joint state is quite simple, but is complicated by two conflicting goals: (i) sequential online construction, and (ii) the need to resolve moments. These complications arise from the existence of 'unresolved links', formed by platform motion and range data.

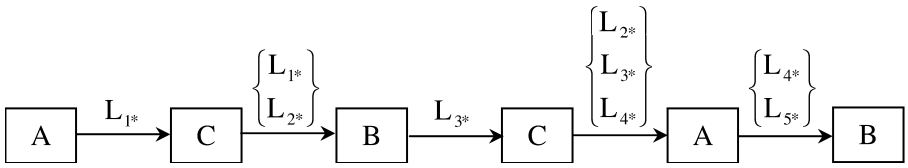


Figure 10: Joint state computation for the first six elements of (25). Note, each $\mathbf{L}_{i*}$ is also accompanied by a $\hat{\mathbf{y}}_i$.

4.3.1. The Essential Process

Operating in the order defined by the augmentation queue, each platform contributes a new state to the joint state vector; adding a new set of elements to $\hat{\mathbf{y}}$ and a new block-row to $\mathbf{L}$. The basic concept is shown in Figure 10 for the ordering in (25). Platform A computes $\{\hat{\mathbf{y}}_1, \mathbf{L}_{1*}\}$ and sends it to platform C. Platform C computes $\{\hat{\mathbf{y}}_2, \mathbf{L}_{2*}\}$ and sends $\{\hat{\mathbf{y}}_{1:2}, \mathbf{L}_{1:2,*}\}$ to platform B. In turn, each platform computes another level of the state vector and forwards it, and all preceding information, to the next platform in the queue. Thus, the information is cumulative, with the i-th platform in the queue holding $\{\hat{\mathbf{y}}_{1:i}, \mathbf{L}_{1:i,*}\}$. (As shown in Figure 10, not all information needs to be communicated each time, since the receiving platform might already know some of the past information.)

Each $\{\hat{\mathbf{y}}_i, \mathbf{L}_{i*}\}$ is produced by the i-th platform to appear in the augmentation queue, and is generated from its local information. For example, at step $i = 4$, platform C sets $\hat{\mathbf{y}}_i = \hat{\mathbf{y}}_{C2}$ and $\mathbf{Y}_{i*} = \mathbf{Y}_{C2,*}$.[2] Given $\mathbf{Y}_{i*}$ and all the previous Cholesky factor rows $\mathbf{L}_{1:i-1,*}$, the new block-row $\mathbf{L}_{i,*}$ is calculated as

$$\mathbf{L}_{1:i-1,1:i-1}\mathbf{L}_{i,1:i-1}^T = \mathbf{Y}_{i,1:i-1}^T \tag{26}$$

$$\mathbf{L}_{i,i}\mathbf{L}_{i,i}^T = \mathbf{Y}_{i,i} - \mathbf{L}_{i,1:i-1}\mathbf{L}_{i,1:i-1}^T \tag{27}$$

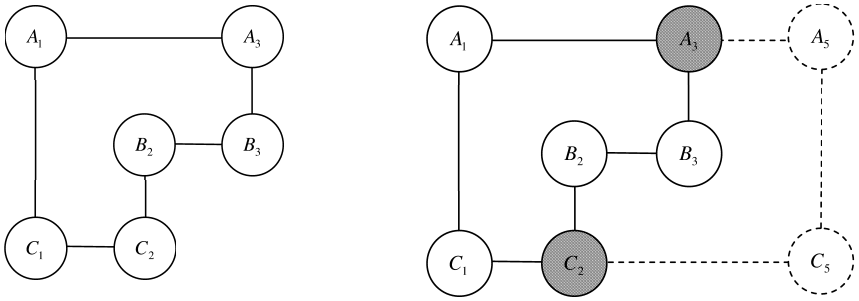
Once a platform has $\{\hat{\mathbf{y}}_{1:i}, \mathbf{L}_{1:i,*}\}$, it is straightforward for it to recover moments (i.e., mean and covariance estimates) for all states according to (18–24). The caveat is that the local estimates used to generate the joint state must have all their links resolved; it is not possible to compute moments when there are unresolved links. This issue is discussed in the next section.

4.3.2. Augmentation and Link Resolution

Consider the two graphs in Figure 11, in particular state C_2, which is the 4-th element in the augmentation queue. Given information up to time-step 3, see Figure 11(a), the state C_2 is linked to C_1 and B_2 only, both of which appear earlier in the augmentation queue, so that the block-row $\mathbf{Y}_{4*}$ corresponding to C_2 is as shown in Figure 12(a).[3] This row has resolved links, since its non-zero terms are all connected to preceding rows. Furthermore, for the graph in Figure 11(a), once the block-row for B_3 has been processed, as in Figure 10, all rows $\mathbf{Y}_{1:6,*}$ have their links resolved, and so platform B can compute the moments of all six states from $\mathbf{L}_{1:6,*}$.

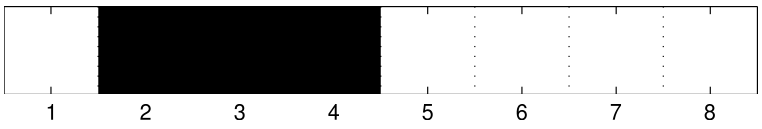
[2] For $\mathbf{Y}_{i*}$ there is a trivial adjustment to place the non-zero elements of $\mathbf{Y}_{C2,*}$ in the correct positions for the joint state vector, as shown in row 4 of Figure 8; no values change, just locations.

[3] Note, the zeros in Figure 12 are just for pictorial purposes. In practice, using a sparse representation, only the non-zero terms of each $\mathbf{Y}_{i*}$ are stored; zero terms are irrelevant.

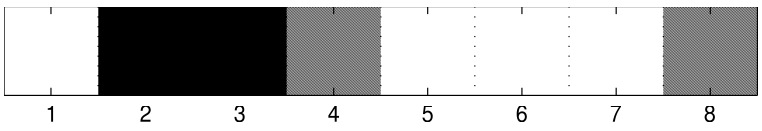


(a) (b)

Figure 11: The joint state given all information up to time-steps (a) 3 and (b) 5. The augmentation steps in (b) add new links and vertices (dashed) and alter the value of connecting vertices (shaded).



(a) Links for Figure 11(a)



(b) Links for Figure 11(b)

Figure 12: Row 4 of $\mathbf{Y}$ for the graphs in Figure 11. The grey shaded squares indicate the values that have changed between (a) and (b) due to augmentation from C_2 to C_5.

Definition. Row i is said to have '***link resolution***' once row j, corresponding to the rightmost non-zero term in row i, has been added to the joint information matrix. □

Unresolved links occur when a row has an off-diagonal term linking it to a *later* row in the augmentation queue. Given information up to time-step 5, there is a new link connecting C_2 to C_5 as shown in Figure 11(b), and the block-row $\mathbf{Y}_{4*}$ now appears as in Figure 12(b). The rightmost non-zero term is in the 8-th column, connecting row 4 to row 8. Thus, link resolution for the revised row 4 is not achieved until row 8 is processed.

Links due to range measurements are always resolved once all platforms with the same time-stamp have been processed, and so are not an issue. Links due to augmentation, however, are a problem because the links never resolve; each row is resolved by a row that itself has an unresolved link. Therefore, to facilitate both augmentation and moment recovery, it is necessary to have two versions of each $\mathbf{Y}_{i*}$. One without augmentation links, for computing moments as in Figure 12(a), and one with augmentation links, for ongoing state construction as in Figure 12(b).

From (26) it can be seen that each new $\mathbf{L}_{i*}$ is a function of the preceding $\mathbf{L}_{1:i-1,*}$, and so any variation in a single $\mathbf{Y}_{i*}$ affects all subsequent rows of the Cholesky factor. Therefore, it is not enough to just have two versions of each $\mathbf{Y}_{i*}$. There needs to be several versions of $\mathbf{L}_{i*}$; one that contains *all* augmentation information for ongoing state construction, and a *series* of variant $\mathbf{L}_{i*}$ rows, generated from $\mathbf{Y}_{i*}$ rows with resolved links, for recovery of selected moments. Computation of the these variant Cholesky factor rows requires a retroactive modification strategy, as described below.

4.3.3. Retroactive State Modification

For this discussion, let row i be the location of the old connecting state (e.g., row 4 for C_2 in the above example) and row j the new augmented state (e.g., row 8 for C_5).

Retroactive augmentation. When the i-th element of the augmentation queue is first processed, and $\mathbf{L}_{i*}$ is added to the joint state, it is computed with a $\mathbf{Y}_{i*}$ that has resolved links (as in Figure 12(a), i.e., no links to row j). This means that $\mathbf{L}_{1:i,*}$ is suitable for moment recovery. For each new augmentation j, processing is revised from the position of the connecting state i in the augmentation queue. The platform first alters $\mathbf{Y}_{i*}$ with the new link information, as in Figure 12(b), then jumps back to the i-th position in the queue. All elements i through j are reprocessed, recomputing $\mathbf{L}_{i:j,*}$. Since the altered $\mathbf{Y}_{i*}$ is resolved at row j, once again $\mathbf{L}_{1:j,*}$ is suitable for moment recovery. The problem with this strategy is that cycling back to the position of the connecting state, and repeating the sequence of computation and inter-platform communications, for each new augmentation is expensive.

Deferred retroactive augmentation. Rather than cycle back to the connecting state i for every new j, more efficient revision is possible by delaying new augmentation of j until the next N elements of the queue are ready. Then augmentation of elements $j : j+N$ is performed as a batch. The cycle begins at position $a \leqslant i$, which is the *earliest* connecting state for the set of new states, and processes $\mathbf{L}_{a:j+N,*}$.

Pipelined retroactive augmentation. While the revision of $\mathbf{L}_{a:j+N,*}$ takes place, there are new elements $j+N+1 : j+N+M$ being added to the augmentation queue. It is not necessary for the second set of augmentations to wait for completion of the first set before starting their own revision cycle. If the earliest connecting state of the second set is b, then its cycle can commence once the revision process of the first set is past position b. For the ordering in (25), suppose that augmentation takes place in three batches:

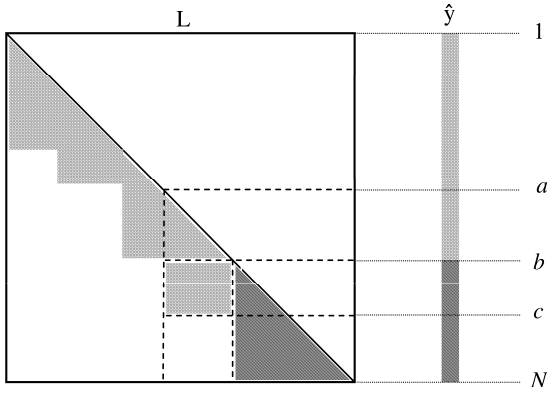


Figure 13: $\mathbf{L}$ and $\hat{\mathbf{y}}$ of the full joint state with elements $1:N$. The most recent states for all platforms are contained in $c:N$. The earliest connection (i.e., leftmost non-zero term) to this set is in column b, and the earliest connection to elements in set $b:c-1$ is in column a.

$\{A_1, C_1, B_2, C_2\}$, $\{A_3, B_3\}$, and $\{A_5, C_5\}$. The second set has its earliest connecting state A_1 at position 1, and processes $\mathbf{L}_{1:6,*}$. The third set has its earliest connecting state C_2 at position 4, and processes $\mathbf{L}_{4:8,*}$. This revision can commence on platform C immediately after the processing for the second set has completed position 4.

4.3.4. Forgetting Past States

Clearly to maintain the full historical joint state is infeasible, since the state grows without bound and the transfer of information between platforms is cumulative. However, in terms of moment recovery, only the most recent states for each platform are of interest, and the historical states are only to permit range fusion and cheap augmentation.

A particular property of the Cholesky factor $\mathbf{L}$ is that it propagates the information from the upper states down to the bottom-right of the matrix. Thus, all information for optimal marginal estimates of the lower states is contained in the bottom-right of $\mathbf{L}$; Cholesky factorisation effects an implicit marginalisation structure with a top-to-bottom elimination order. Therefore, to recover optimal moments of lower states, the upper portions of $\mathbf{L}$ are not required.

Consider the joint state in Figure 13, where the *most recent states for all platforms* are contained in the lowest elements $c:N$. If the earliest connecting state to this set is element b then, according to the properties in Section 3.4, the act of adding $c:N$ to the joint state only modifies $\{\hat{\mathbf{y}}_{b:N}, \mathbf{Y}_{b:N,b:N}, \mathbf{L}_{b:N,b:N}\}$ as shown in Figure 14. The retroactive modification of $\mathbf{L}_{b:c-1,*}$ is computed as

$$\mathbf{L}_{b:i-1,b:i-1}\mathbf{L}_{i,b:i-1}^T = \mathbf{Y}_{i,b:i-1}^T - \mathbf{L}_{b:i-1,a:b-1}\mathbf{L}_{i,a:b-1}^T, \tag{28}$$

$$\mathbf{L}_{i,i}\mathbf{L}_{i,i}^T = \mathbf{Y}_{i,i} - \mathbf{L}_{i,a:i-1}\mathbf{L}_{i,a:i-1}^T, \tag{29}$$

where $b \leqslant i < c$. The terms to the left of column b in $\mathbf{L}_{b:c-1,*}$ are unchanged and do not have to be recomputed. For the new augmented states $\mathbf{L}_{c:N,*}$, the block-rows $c \leqslant i \leqslant N$ are computed as

$$\mathbf{L}_{b:i-1,b:i-1}\mathbf{L}_{i,b:i-1}^T = \mathbf{Y}_{i,b:i-1}^T, \tag{30}$$

$$\mathbf{L}_{i,i}\mathbf{L}_{i,i}^T = \mathbf{Y}_{i,i} - \mathbf{L}_{i,b:i-1}\mathbf{L}_{i,b:i-1}^T. \tag{31}$$

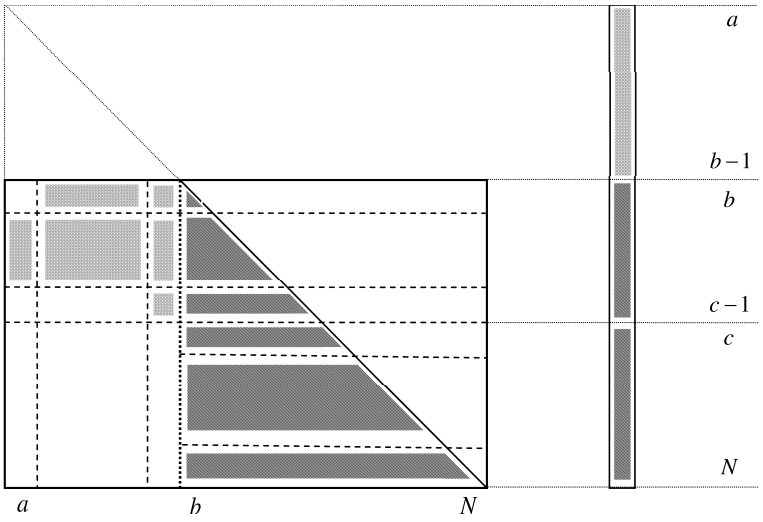


Figure 14: Detailed view of the relevant submatrix and subvector from Figure 13, showing the key block-rows of a, b, c and N. The act of adding elements $c:N$ to the joint state modifies $\hat{\mathbf{y}}_{b:N}$ and $\mathbf{L}_{b:N,b:N}$. It also references, but does not modify, $\hat{\mathbf{f}}_{a:b-1}$ and $\mathbf{L}_{b:c-1,a:b-1}$. The state elements $1:a-1$ are obsolete and can be forgotten.

Recovery of moments $\{\hat{\mathbf{x}}_{b:N}, \mathbf{P}_{b:N,b:N}\}$ is straightforward from Sections 3.4.2 and 3.4.3. It is worth noting that for the mean calculation,

$$\mathbf{L}_{b:N,b:N}\hat{\mathbf{f}}_{b:N} = \hat{\mathbf{y}}_{b:N} - \mathbf{L}_{b:c-1,a:b-1}\hat{\mathbf{f}}_{a:b-1}, \tag{32}$$

the value of $\hat{\mathbf{f}}_{a:b-1}$ is not modified by the augmentation of elements $c:N$, and can be reused from when it was computed during moment recovery of these earlier states.

From the above equations, it is clear that the rows of $\{\hat{\mathbf{y}}, \mathbf{Y}, \mathbf{L}\}$ above element b are not required for either future Cholesky modifications or moment recovery of states b onwards. (The one exception is the need to retain $\hat{\mathbf{f}}_{a:b-1}$.) Since these parts are obsolete, they do not have to be stored or communicated, and can be deleted, keeping only the submatrix and subvector shown in Figure 14. This bounds computation and communication costs.

5. Experiments

The decentralised solution proposed in this paper is tested on a series of simulation experiments. Ten mobile platforms (labelled with A, B, C, etc), each moving according to a random walk behavior, make range observations to each other and can communicate with each other at any time. Only Platform A has GPS receiver, which allows us to demonstrate the propagation of position information to other platforms (that lack GPS) via inter-platform range measurements. The experimental parameters are as follows. The standard deviations are: initial platform positions, 5 metres; velocity measurements (from odometry), 1 metre/sec; GPS measurements, 5 metres; range measurements, 2 metres. The system operates in $\Delta T = 0.1s$ time steps. At each estimation time step, the probabilities of a platform obtaining a measurement are: GPS, 10% (Platform A only); range measurement to another platform, 5%. Note, the probability for range measurements is for *each* platform interaction; so each time step, Platform A has 5%

probability of measuring Platform B, and 5% probability of measuring Platform C, etc. Among all the parameters, the initial position error is chosen be small enough to mitigate the problem of large non-linearities, and the other parameter values are non-critical.

Our simulated system has time-synchronous measurements. However, the decentralised algorithm applies equally to real-world asynchronous systems, in which case time-alignment is performed by projecting forward the platform states (according to the motion model) to match the observation timestamps. This is the same approach as is routinely applied in centralised filtering systems.

5.1. Comparison with a Distributed Solution

The distributed fusion algorithm presented in [16] was the basis for this current work. The algorithm in [16] is distributed in the sense that information from each platform is shared across the network, but is *not* decentralised because each platform accumulates the full joint state history. Thus, every platform acts as a fusion centre and duplicates the same joint estimate. The solution presented in this paper shares many of the local estimation operations as in [16]: each platform builds a local Markov chain estimate from odometry and GPS, and removes irrelevant states (i.e., states not involved in range measurements).

The key difference is the formation of the joint state. In the current paper, range measurements are shared only between pairs of platform and fused locally, and the joint state is constructed by sequential communication according to the augmentation queue. No single platform has the entire historical joint state (although a single platform *will* have sufficient joint state to compute the most recent estimates of all platforms). In [16], on the other hand, all range measurements are transmitted to all platforms, and each platform constructs the *full* joint state history. Communication between platforms is ad hoc, rather than sequential.

In the experiments that follow, a variation on [16] is implemented where all ranges and platform states are sent to a *single* fusion centre. This avoids the redundancy of the original solution, which makes every platform an identical fusion centre.

5.2. Results

Each platform in the team exhibits different estimation accuracies and time lags. In the following figures, we show the results for the x-axis position of platform C. It is important to realise that the estimators described here are optimal, in the same sense as a Kalman smoother, but exhibit lags due to communication delays. However, each platform is able to compute a suboptimal estimate of its own current state by predicting forward from the most recent optimal estimate using its odometry. Figure 15 shows the lagged optimal estimate and the 'real-time' suboptimal estimate. The time lags for the optimal estimates are shown in Figure 16.

The estimation time lag of the decentralised method is larger than that in the distributed method as shown in Figure 16. The main delay in the decentralised method is caused by the sequential processing of the augmentation queue to perform distributed Cholesky modification.

The estimation accuracy should be evaluated on the whole platform team, not on a single platform. We define the root mean square error ($rmse$) of x-axis position estimation for the platform team as follows:

$$rmse(k) = \sqrt{\frac{\sum_{i \in N_k} (\hat{x}_k^i - x_k^i)^2}{|N_k|}} \tag{33}$$

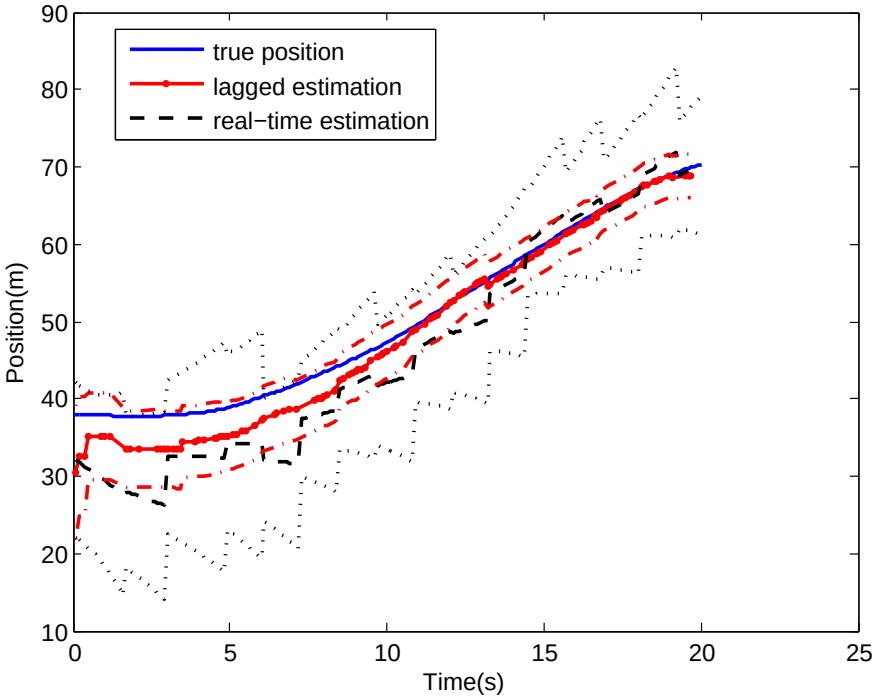


Figure 15: The lagged and current-time estimates and 2-σ bounds for the x-axis position of Platform C. Each delayed estimate is shown here at its 'real' time; the optimal estimate $\hat{\mathbf{x}}_k$ is shown at time k, even though it is not available until time $k + t_{lag}$. Instead, the lag is shown in Figure 16.

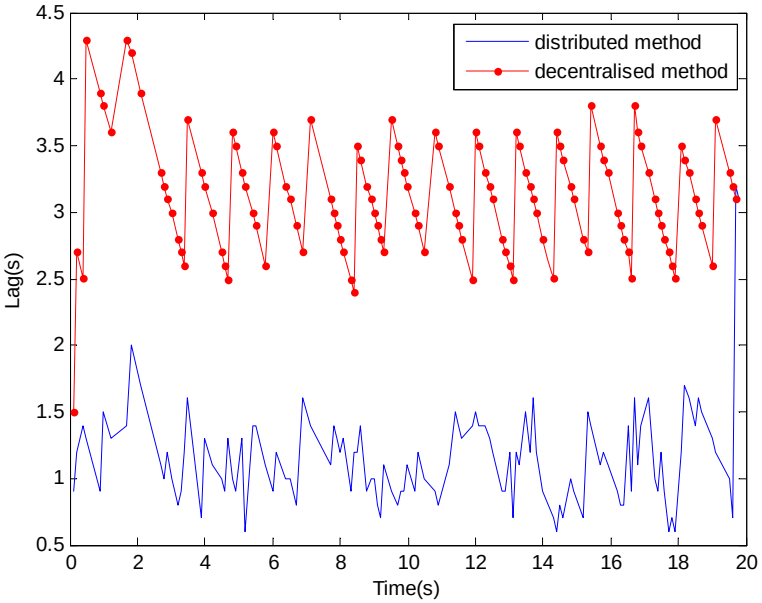


Figure 16: Time lags of the decentralised and distributed method. The state at each moment k is estimated at time $k + t_{lag}$.

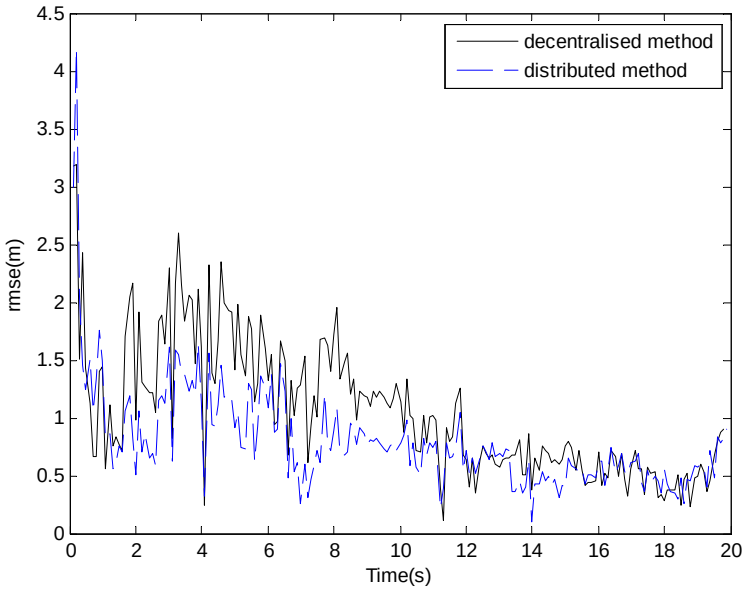


Figure 17: Cooperative navigation accuracy of the distributed method and the decentralised method. Note, although the delayed (optimal) estimates from the two methods are of the same accuracy, the current-time (predictive) estimates from the distributed method is more accurate due to smaller time lags.

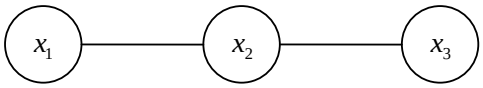


Figure 18: Graphical model of the three basic parts of the joint state: the obsolete historical states $\mathbf{x}_1$, the link states $\mathbf{x}_2$, and the current states $\mathbf{x}_3$.

where x_k^i is the ground truth value of x-axis position at time k for Platform i, $\hat{x}_k^i$ is the corresponding estimate; N_k is the set of platforms whose state at time k is recovered. The results of $rmse$ of the distributed method and the decentralised method are shown in Figure 17.

6. Cost Analysis

We compare the decentralised method with the distributed method in terms of computation cost, communication cost and robustness to platform failure.

6.1. Computational Costs

The computational costs of *local* operations—processing odometry and GPS—are essentially the same for both methods. Markov chain construction is done on each platform independently, and so the per-platform cost is independent of the size of the team. Odometry data augmentation and GPS data fusion operations, each involve $O(n^2)$ computational cost, where n is the dimension of an instantaneous state for a platform.

The computational costs involved in moment recovery are different for the two methods. Consider the representative system in (Figure 18) composed of three parts. The

Table 1: comparison of the moment recovery costs

method	operation	cost
distributed covariance	$\tilde{\mathbf{Y}}_{22} = \mathbf{Y}_{22} - \mathbf{Y}_{12}^T \mathbf{Y}_{11}^{-1} \mathbf{Y}_{12}$	$\mathbf{O}(k_1^3 + k_1^2 k_2 + k_2^2 k_1)$
	$\mathbf{P} = \tilde{\mathbf{Y}}^{-1}$, where $\tilde{\mathbf{Y}} = \begin{bmatrix} \tilde{\mathbf{Y}}_{22} & \mathbf{Y}_{23} \\ \mathbf{Y}_{23}^T & \mathbf{Y}_{33} \end{bmatrix}$	$\mathbf{O}(k_2 + k_3)^3$
distributed mean	$\tilde{\mathbf{y}}_2 = \mathbf{y}_2 - \mathbf{Y}_{12}^T \mathbf{Y}_{11}^{-1} \mathbf{y}_1$	$\mathbf{O}(k_1 k_2)$
	$\mathbf{x} = \mathbf{P}\tilde{\mathbf{y}}$, where $\tilde{\mathbf{y}} = \begin{bmatrix} \tilde{\mathbf{y}}_2^T & \tilde{\mathbf{y}}_3^T \end{bmatrix}^T$	$\mathbf{O}(k_2 + k_3)^2$
decentralised covariance	$\tilde{\mathbf{L}}_{22}\tilde{\mathbf{L}}_{22}^T = \tilde{\mathbf{Y}}_{22} - \mathbf{L}_{21}\mathbf{L}_{21}^T$	$\mathbf{O}(k_2^2 k_1 + k_2^3)$
	$\tilde{\mathbf{L}}_{22}\mathbf{L}_{32}^T = \mathbf{Y}_{32}^T$	$\mathbf{O}(k_2^2 k_3)$
	$\mathbf{L}_{33}\mathbf{L}_{33}^T = \mathbf{Y}_{33} - \mathbf{L}_{32}\mathbf{L}_{32}^T$	$\mathbf{O}(k_3^2 k_2 + k_3^3)$
	Equation (24), where $\mathbf{L} = \begin{bmatrix} \tilde{\mathbf{L}}_{22} & \mathbf{0} \\ \mathbf{L}_{32} & \mathbf{L}_{33} \end{bmatrix}$	$\mathbf{O}(k_2 + k_3)^3$
decentralised mean	$\tilde{\mathbf{L}}_{22}\tilde{\mathbf{f}}_2 = \mathbf{y}_2 - \mathbf{L}_{21}\mathbf{f}_1$	$\mathbf{O}(k_2^2) + \mathbf{O}(k_1 k_2)$
	$\mathbf{L}_{33}\mathbf{f}_3 = \mathbf{y}_3 - \mathbf{L}_{32}\tilde{\mathbf{f}}_2$	$\mathbf{O}(k_3^2) + \mathbf{O}(k_2 k_3)$
	$\mathbf{L}^T\mathbf{x} = \begin{bmatrix} \tilde{\mathbf{f}}_2^T & \mathbf{f}_3^T \end{bmatrix}^T$	$\mathbf{O}(k_2 + k_3)^2$

historical (i.e., older, obsolete) states are $\mathbf{x}_1$, the states $\mathbf{x}_2$ are the set of linking states, and $\mathbf{x}_3$ are the set of current states. For a system of N platforms, there are $O(Nn)$ states in $\mathbf{x}_3$, and a similar number of linking states in $\mathbf{x}_2$. The great majority of states are in $\mathbf{x}_1$. Over time, the information submatrix for $\mathbf{x}_3$ becomes dense, as does its connections to the linking states, and so the computational costs of moment recovery may be considered a function of the length of these subvectors. The operations and computational costs to recover the moments for states $\mathbf{x} = \begin{bmatrix} \mathbf{x}_2^T, \mathbf{x}_3^T \end{bmatrix}^T$ with the distributed and decentralised methods are listed in Table 1, where k_i is the length of vector $\mathbf{x}_i$ for $i = 1, 2, 3$. These values show that the computational costs for the decentralised method are comparable to that of the distributed method.

6.2. Communication Costs

The communication costs for the distributed and decentralised methods are quite different, since the first performs joint estimation and marginalisation at a single fusion centre while the latter distributes the joint state. For distributed method, the main communication cost is the transmission of all range measurements to the server, each accompanied by a pair of instantaneous vehicle-position states. This cost is $O(Nn^2)$ per-timestep, where n is the length of a platform's instantaneous state vector and N is the number of platforms.

For the distributed method the cost is considerably higher, since the accumulated set of Cholesky factors rows are forwarded in augmentation queue sequence, as shown in Figure 10. Since for $i \gg 1$ the number of non-zero terms in each $\mathbf{L}_{i*}$ is $O(Nn^2)$, for a system of N platforms the per-timestep communications cost is $O(N^2 n^2)$. The cost will be higher still due to the repetition of communications with retroactive augmentation.

6.3. Robustness to Platform Failure

Suppose a single platform fails and ceases to communicate. We assume this failure is detectable by the other platforms. In this case, both distributed and decentralised methods are able to operate consistently. The distributed system in Section 5.1 is unaffected unless the failed platform is the server platform. Server failure is addressed simply by having multiple redundant fusion centres. The extreme case described in [16] has every platform a server, but good robustness is achievable with far less redundancy.

For the decentralised solution described in this paper, redundancy also exists on every platform, and recovery to platform failure is accomplished by modifying the augmentation queue to eliminate future communication with the lost platform. This operation is complicated by the existence of unresolved links to the platform, whether due to range measurements or motion measurements. So, recovery involves eliminating the failed platform from the augmentation queue and removing any unresolved links to that platform.

7. Conclusions and Extensions

In the paper, a novel decentralised approach to the cooperative multi-platform navigation problem is investigated. The approach uses the information filter framework and integrates delayed states and distributed Cholesky modification. In the decentralised solution, localised fusion distributes the computation load and avoids transmission of high frequency motion data; the modular fusion capability of each platform provides good scalability and robustness to platform failure. The estimates obtained are optimal, in the same sense as a Kalman smoother, but are lagged due to communication delays. Suboptimal predictions from the most recent (but lagged) optimal solution permits real-time estimates of the current state.

The scenario given in Section 3.1 describes a general cooperative navigation problem that can be solved using the decentralised approach. In the scenario, a variety of sensing modalities can be employed; the platform team can be both homogeneous and heterogeneous. Further more, the approach is not limited to the cooperative navigation problem, it can be applied to cooperative target tracking, cooperative exploration, cooperative SLAM(simultaneous localization and mapping), etc. The decentralised structure appears to be a general paradigm for decentralised estimation.

Comparing with the distributed solution presented in [16], the decentralised solution generates the same optimal result, but with a very different distribution strategy. It has similar costs but is not a superior algorithm. Its estimation lag is larger and it is more expensive in terms of communication. In sum, the distribution and communication of Cholesky factors is more expensive than marginalisation and moment recovery from information states at a fusion centre.

In future work, we plan to investigate the effects of linearisation on decentralised estimation. Linear model approximations are routinely applied to nonlinear systems, whether opportunistically, as with an EKF, or with iterative revision, as with iterated Kalman smoothing. The problems of opportunistic linearisation is exacerbated in a distributed system, since approximations may be based on less available information due to communication and estimation lags. Whether an efficient means of iterative or revised linearisation is possible for decentralised estimation is of significant interest.

Appendix A: State and Information Operations

This appendix briefly summarises state and information form parameterisations of multi-variate Gaussians. The key operations of marginalisation and conditioning in each form are presented.

A.1. Covariance and Information Parameterisations for Gaussians

Consider a random variable $\mathbf{x}$ whose probability density function (pdf) is Gaussian. This pdf can be parameterised either in terms of a state mean $\hat{\mathbf{x}}$ and variance $\mathbf{P}$ as

$$P(\mathbf{x}) = \mathcal{N}_S(\mathbf{x}; \hat{\mathbf{x}}, \mathbf{P}) \propto \exp\left(-\frac{1}{2}\left(\mathbf{x}-\hat{\mathbf{x}}\right)^T \mathbf{P}^{-1}\left(\mathbf{x}-\hat{\mathbf{x}}\right)\right) \tag{A-1}$$

or in terms of an information vector $\hat{\mathbf{y}}$ and information matrix $\mathbf{Y}$ in *information form* (sometimes called *canonical form*) as

$$P(\mathbf{x}) = \mathcal{N}_I(\mathbf{x}; \hat{\mathbf{y}}, \mathbf{Y}) \propto \exp\left(-\frac{1}{2}\mathbf{x}^T\mathbf{Y}\mathbf{x} + \mathbf{x}^T\hat{\mathbf{y}}\right) \tag{A-2}$$

where

$$\mathbf{Y} = \mathbf{P}^{-1}, \qquad \text{and} \qquad \hat{\mathbf{y}} = \mathbf{P}^{-1}\hat{\mathbf{x}}. \tag{A-3}$$

Consider now a joint state-vector $\mathbf{x} = \left[\mathbf{x}_1^T, \mathbf{x}_2^T\right]^T$, the joint pdf may be written either in the form

$$P(\mathbf{x}_1, \mathbf{x}_2) \propto \exp\left(-\frac{1}{2}\begin{bmatrix} \mathbf{x}_1 - \hat{\mathbf{x}}_1 \\ \mathbf{x}_2 - \hat{\mathbf{x}}_2 \end{bmatrix}^T \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{12}^T & \mathbf{P}_{22} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{x}_1 - \hat{\mathbf{x}}_1 \\ \mathbf{x}_2 - \hat{\mathbf{x}}_2 \end{bmatrix}\right) \tag{A-4}$$

or as

$$P(\mathbf{x}_1, \mathbf{x}_2) \propto \exp\left(-\frac{1}{2}\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}^T \begin{bmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \\ \mathbf{Y}_{12}^T & \mathbf{Y}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}^T \begin{bmatrix} \hat{\mathbf{y}}_1 \\ \hat{\mathbf{y}}_2 \end{bmatrix}\right) \tag{A-5}$$

where

$$\begin{bmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \\ \mathbf{Y}_{12}^T & \mathbf{Y}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{12}^T & \mathbf{P}_{22} \end{bmatrix}^{-1} \tag{A-6}$$

and

$$\begin{aligned} \begin{bmatrix} \hat{\mathbf{y}}_1 \\ \hat{\mathbf{y}}_2 \end{bmatrix} &= \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{12}^T & \mathbf{P}_{22} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \\ \mathbf{Y}_{12}^T & \mathbf{Y}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix}. \end{aligned} \tag{A-7}$$

It should be noted that $\mathbf{Y}_{ii} \neq \mathbf{P}_{ii}^{-1}$, $i = 1, 2$.

A.2. Marginalisation

Marginalisation eliminates one of the two variables through integration

$$P(\mathbf{x}_1) = \int P(\mathbf{x}_1, \mathbf{x}_2)\mathrm{d}\mathbf{x}_2, \qquad P(\mathbf{x}_2) = \int P(\mathbf{x}_1, \mathbf{x}_2)\mathrm{d}\mathbf{x}_1. \tag{A-8}$$

In covariance form marginalisation is simply variable elimination, so,

$$P(\mathbf{x}_1) = \mathcal{N}_S(\mathbf{x}_1; \hat{\mathbf{x}}_1, \mathbf{P}_{11}), \qquad P(\mathbf{x}_2) = \mathcal{N}_S(\mathbf{x}_2; \hat{\mathbf{x}}_2, \mathbf{P}_{22}). \tag{A-9}$$

In information form we have

$$P(\mathbf{x}_1) = \mathcal{N}_I(\mathbf{x}_1; \tilde{\mathbf{y}}_1, \tilde{\mathbf{Y}}_1), \qquad P(\mathbf{x}_2) = \mathcal{N}_I(\mathbf{x}_2; \tilde{\mathbf{y}}_2, \tilde{\mathbf{Y}}_2), \tag{A-10}$$

where the parameters of the marginal densities are given by

$$\begin{aligned} \tilde{\mathbf{y}}_1 &= \hat{\mathbf{y}}_1 - \mathbf{Y}_{12}\mathbf{Y}_{22}^{-1}\hat{\mathbf{y}}_2, & \tilde{\mathbf{Y}}_1 &= \mathbf{Y}_{11} - \mathbf{Y}_{12}\mathbf{Y}_{22}^{-1}\mathbf{Y}_{12}^T, \\ \tilde{\mathbf{y}}_2 &= \hat{\mathbf{y}}_2 - \mathbf{Y}_{12}^T\mathbf{Y}_{11}^{-1}\hat{\mathbf{y}}_1, & \tilde{\mathbf{Y}}_2 &= \mathbf{Y}_{22} - \mathbf{Y}_{12}^T\mathbf{Y}_{11}^{-1}\mathbf{Y}_{12}, \end{aligned} \tag{A-11}$$

and where, in contrast to (A-7), $\tilde{\mathbf{Y}}_i = \mathbf{P}_{ii}^{-1}$, $i = 1, 2$.

A.3. Conditioning

Recall the definition of conditional probability,

$$P(\mathbf{x}_1 \mid \mathbf{x}_2) = \frac{P(\mathbf{x}_1, \mathbf{x}_2)}{P(\mathbf{x}_2)}, \qquad \text{and} \quad P(\mathbf{x}_2 \mid \mathbf{x}_1) = \frac{P(\mathbf{x}_1, \mathbf{x}_2)}{P(\mathbf{x}_1)}. \tag{A-12}$$

In covariance form, we have

$$P(\mathbf{x}_1 \mid \mathbf{x}_2 = x_2) = \mathcal{N}_S(\mathbf{x}_1; \hat{\mathbf{x}}_{1|2}, \mathbf{P}_{1|2}), \qquad P(\mathbf{x}_2 \mid \mathbf{x}_1 = x_1) = \mathcal{N}_S(\mathbf{x}_2; \hat{\mathbf{x}}_{2|1}, \mathbf{P}_{2|1}) \tag{A-13}$$

where

$$\begin{aligned} \hat{\mathbf{x}}_{1|2} &= \hat{\mathbf{x}}_1 + \mathbf{P}_{12}\mathbf{P}_{22}^{-1}(x_2 - \hat{\mathbf{x}}_2) \\ \mathbf{P}_{1|2} &= \mathbf{P}_{11} - \mathbf{P}_{12}\mathbf{P}_{22}^{-1}\mathbf{P}_{12}^T, \end{aligned} \tag{A-14}$$

$$\begin{aligned} \hat{\mathbf{x}}_{2|1} &= \hat{\mathbf{x}}_2 + \mathbf{P}_{21}\mathbf{P}_{11}^{-1}(x_1 - \hat{\mathbf{x}}_1) \\ \mathbf{P}_{2|1} &= \mathbf{P}_{22} - \mathbf{P}_{21}\mathbf{P}_{11}^{-1}\mathbf{P}_{21}^T. \end{aligned} \tag{A-15}$$

In information form the conditional can be found by variable elimination in an analogous manner to that for state-variable marginalisation

$$P(\mathbf{x}_1 \mid \mathbf{x}_2 = x_2) = \mathcal{N}_I(\mathbf{x}_1; \hat{\mathbf{y}}_{1|2}, \mathbf{Y}_{11}), \qquad P(\mathbf{x}_2 \mid \mathbf{x}_1 = x_1) = \mathcal{N}_I(\mathbf{x}_2; \hat{\mathbf{y}}_{2|1}, \mathbf{Y}_{22}), \tag{A-16}$$

where $\mathbf{Y}_{11}$ and $\mathbf{Y}_{22}$ are the sub-information matrices taken directly from the joint, and the information vectors are given by

$$\hat{\mathbf{y}}_{1|2} = \hat{\mathbf{y}}_1 - \mathbf{Y}_{12}x_2, \qquad \hat{\mathbf{y}}_{2|1} = \hat{\mathbf{y}}_2 - \mathbf{Y}_{12}^T x_1. \tag{A-17}$$

A.4. Linear Relations Between Variates

Consider a random variable $\mathbf{x}_2$ constructed from a linear combination of two other random variables $\mathbf{x}_1$ and $\mathbf{u}$ as

$$\mathbf{x}_2 = \mathbf{T}\mathbf{x}_1 + \mathbf{u}, \qquad P(\mathbf{x}_1) = \mathcal{N}_S(\mathbf{x}_1; \hat{\mathbf{x}}_1, \mathbf{P}_1), \quad P(\mathbf{u}) \sim \mathcal{N}_S(\mathbf{u}; \hat{\mathbf{u}}, \mathbf{U}). \tag{A-18}$$

The mean and variance of the joint $P(\mathbf{x}_1, \mathbf{x}_2)$ are given by

$$\begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{12}^T & \mathbf{P}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{11}\mathbf{T}^T \\ \mathbf{T}\mathbf{P}_{11} & \mathbf{T}\mathbf{P}_{11}\mathbf{T}^T + \mathbf{U} \end{bmatrix}, \quad \begin{bmatrix} \hat{\mathbf{x}}_1 \\ \hat{\mathbf{x}}_2 \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_1 \\ \mathbf{T}\hat{\mathbf{x}}_1 + \hat{\mathbf{u}} \end{bmatrix}. \tag{A-19}$$

The corresponding information matrix can be obtained by direct inversion of the joint covariance matrix

$$\begin{bmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \\ \mathbf{Y}_{12}^T & \mathbf{Y}_{22} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{12}^T & \mathbf{P}_{22} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{P}_{11}^{-1} + \mathbf{T}^T\mathbf{U}^{-1}\mathbf{T} & -\mathbf{T}^T\mathbf{U}^{-1} \\ -\mathbf{U}^{-1}\mathbf{T} & \mathbf{U}^{-1} \end{bmatrix}. \tag{A-20}$$

The joint information vector can now be found as

$$\begin{bmatrix} \hat{\mathbf{y}}_1 \\ \hat{\mathbf{y}}_2 \end{bmatrix} = \mathbf{Y}\begin{bmatrix} \hat{\mathbf{x}}_1 \\ \hat{\mathbf{x}}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{11}^{-1}\hat{\mathbf{x}}_1 - \mathbf{T}^T\mathbf{U}^{-1}\hat{\mathbf{u}} \\ \mathbf{U}^{-1}\hat{\mathbf{u}} \end{bmatrix}. \tag{A-21}$$

A.5. Observation Updates

Given a prior probability of joint state-vector $\mathbf{x} = \left[\mathbf{x}_1^T, \mathbf{x}_2^T\right]^T$ as in (A-4) or (A-5), and an observation of the state $\mathbf{x}_2$,

$$\mathbf{z} = \mathbf{H}\mathbf{x}_2 + \mathbf{r}, \qquad \mathbf{r} \sim \mathcal{N}_S(\mathbf{r}; \mathbf{0}, \mathbf{R}), \tag{A-22}$$

the fusion of this observation with the prior is accomplished by conditioning as

$$P(\mathbf{x}_1, \mathbf{x}_2 \mid \mathbf{z}) = \frac{P(\mathbf{x}_1, \mathbf{x}_2, \mathbf{z})}{P(\mathbf{z})}, \qquad P(\mathbf{z}) = \int\int P(\mathbf{x}_1, \mathbf{x}_2, \mathbf{z}) \mathrm{d}\mathbf{x}_1 \mathrm{d}\mathbf{x}_2. \tag{A-23}$$

From Equations (A-19, A-14), we have $P(\mathbf{x} \mid \mathbf{z} = z) = \mathcal{N}_S\left(\mathbf{x}; \hat{\mathbf{x}}_{|z}, \mathbf{P}_{|z}\right)$, where

$$\begin{aligned} \hat{\mathbf{x}}_{|z} &= \begin{bmatrix} \hat{\mathbf{x}}_1 \\ \hat{\mathbf{x}}_2 \end{bmatrix} + \begin{bmatrix} \mathbf{P}_{12}\mathbf{H}^T \\ \mathbf{P}_{22}\mathbf{H}^T \end{bmatrix} \left(\mathbf{H}\mathbf{P}_{22}\mathbf{H}^T + \mathbf{R}\right)^{-1} \left(z - \mathbf{H}\hat{\mathbf{x}}_2\right), \\ \mathbf{P}_{|z} &= \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{12}^T & \mathbf{P}_{22} \end{bmatrix} - \begin{bmatrix} \mathbf{P}_{12}\mathbf{H}^T \\ \mathbf{P}_{22}\mathbf{H}^T \end{bmatrix} \left(\mathbf{H}\mathbf{P}_{22}\mathbf{H}^T + \mathbf{R}\right)^{-1} \begin{bmatrix} \mathbf{H}\mathbf{P}_{12}^T & \mathbf{H}\mathbf{P}_{22} \end{bmatrix}. \end{aligned} \tag{A-24}$$

From Equations (A-20,A-21, A-16,A-17), we have $P(\mathbf{x} \mid \mathbf{z} = z) = \mathcal{N}_I\left(\mathbf{x}; \hat{\mathbf{y}}_{|z}, \mathbf{Y}_{|z}\right)$, where

$$\hat{\mathbf{y}}_{|z} = \begin{bmatrix} \hat{\mathbf{y}}_1 \\ \hat{\mathbf{y}}_2 + \mathbf{H}^T\mathbf{R}^{-1}z \end{bmatrix}, \quad \mathbf{Y}_{|z} = \begin{bmatrix} \mathbf{Y}_{11} & \mathbf{Y}_{12} \\ \mathbf{Y}_{12}^T & \mathbf{Y}_{22} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} \end{bmatrix}. \tag{A-25}$$

In information form, fusion modifies only the sub-states involved in the observation model.

References

[1] L. E. Parker, "Current state of the art in distributed autonomous mobile robotics," in *Distributed Autonomous Robotic System 6*, R. Alami, R. Chatila, and H. Asama, Eds. Springer-Verlag, Tokyo, October 2000, pp. 3–12.

[2] P. Maybeck, *Stochastic Models, Estimation and Control*. Academic Press, New York, 1979, vol. 1.

[3] Y. Saad, *Iterative methods for sparse linear systems*, 2nd ed. SIAM, 2004.

[4] S. Roumeliotis and G. Bekey, "Distributed multi-robot localization," *IEEE Transactions on Robotics and Automation*, vol. 18, pp. 781–795, 2002.

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

[6] A. Howard, M. Matari, and G. Sukhatme, "Localization for mobile robot teams: A distributed MLE approach," in *Experimental Robotics VIII*, B. Siciliano and P. Dario, Eds. Springer-Verlag, 2003, pp. 146–155.

[7] R. Madhavan, K. Fregene, and L. Parker, "Distributed heterogeneous outdoor multi-robot localization," in *the IEEE International Conference on Robotics and Automation*, 2002, pp. 374–381.

[8] S. Thrun, Y.Liu, D. Koller, A. Ng, and H. Durrant-Whyte, "Simultaneous localisation and mapping with sparse extended information filters," *International Journal of Robotics Research*, vol. 23, no. 7–8, pp. 693–716, 2004.

[9] R. M. Eustice, H. Singh, and J. J. Leonard, "Exactly sparse delayed-state filters," in *IEEE International Conference on Robotics and Automation*, 2005, pp. 2417–2424.

[10] F. Dellaert and M. Kaess, "Square root SAM: Simultaneous localization and mapping via square root information smoothing," *International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, 2006.

[11] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (SLAM): Part II," *IEEE Robotics and Automation Magazine*, vol. 13, no. 3, pp. 108–117, 2006.

[12] M. Deans and M. Hebert, "Experimental comparison of techniques for localization and mapping using a bearing-only sensor," in *Experimental Robotics VII*, D. Rus and S. Singh, Eds. Springer-Verlag, 2000, vol. 271, pp. 395–404.

[13] J. J. Leonard and R. J. Rikoski, "Incorporation of delayed decision making into stochastic mapping," in *Experimental Robotics VII*, D. Rus and S. Singh, Eds. Springer-Verlag, 2001, vol. 271, pp. 533–542.

[14] J. Leonard, R. Rikoski, P. Newman, and M. Bosse, "Mapping partially observable features from multiple uncertain vantage points," *The International Journal of Robotics Research*, vol. 21, no. 10–11, pp. 943–975, 2002.

[15] I. Mahon, S. Williams, O. Pizarro, and M. Johnson-Roberson, "Efficient view-based SLAM using visual loop closures," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1002–1014, 2008.

[16] T. Bailey and H. Durrant-Whyte, "Decentralised data fusion with delayed states for consistent inference in mobile ad hoc networks," 2007. [Online]. Available: http://www-personal.acfr.usyd.edu.au/tbailey/

[17] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd ed. Johns Hopkins University Press, 1996.

[18] Y. U. Cao, A. S. Fukunaga, and A. B. Kahng, "Cooperative mobile robotics: Antecedents and directions," *Autonomous Robots*, vol. 4, no. 1, pp. 7–27, March 1997.

[19] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[20] A. Björck, *Numerical Methods for Least Squares Problems*. SIAM, 1996.