A Server-Client Based Distributed Processing for an Unscented Kalman Filter for Cooperative Localization

Vu Dinh and Solmaz S. Kia

Abstract— **For a group of mobile robots with communication and computation capabilities, we consider a cooperative localization algorithm based on Unscented Kalman filtering. We present a server-client paradigm to distribute the computational cost of this algorithm among team members. The highest computational cost of the Unscented Kalamn filter comes from calculating the collective covariance matrix of the team and its square root, normally obtained by Cholesky decomposition. Our server-client based computationally distributed algorithm is centered on identifying an appropriate Cholesky decomposition algorithm which allows a coordinated computational task allocation among team members.**

I. Introduction

Accurate localization is crucial for the success of mobile multi-robot operations such as search and rescue, environmental monitoring, and oceanic exploration. These operations most of the time take place in fully or partially GPS-denied environments. Moreover, often, the environment is a priori unaccessible, without distinct features, or the features are changing with time. As a result, GPS navigation, classical beacon-based localization algorithms [1] or fixed feature-based Simultaneous Localization and Mapping algorithms [2] cannot be used. The *Cooperative Localization* (CL) strategy for multi-robot systems is a technique with a promising prospect that uses relative measurements among the robots as a feedback signal to jointly estimate the location of team members, resulting in improved position accuracy for the entire team. However, because CL treats localization as a joint estimation problem, any algorithm applied results in a significant processing requirements whose reduction has been the subject of research over the past few years (see e.g., [3], [4], [5], [6]). In this paper, we intend to distribute the high computational cost of a centralized Unscented Kalamn filter (UKF) algorithm for CL among team members using a server-client paradigm.

In recent years, many effort have been devoted to develop decentralized CL (D-CL) algorithms where the high communication and computation costs of the CL strategy are distributed among the team members. For example, [9] proposes a suboptimal algorithm that only the robot obtaining the relative measurement updates its states. There, to produce consistent estimates, a bank of Extended Kalman Filters (EKFs) is maintained at each robot. Using an accurate bookkeeping of the identity of the robots involved in previous updates, each filter is only updated when its states are not correlated with the state of the landmark robot. The main drawbacks of this algorithm is the large memory demand, computational complexity, and the growing size of information needed at each update. An alternative approach is to distribute the computation of components of a centralized CL among team members. This decentralization can be conducted as a multi-centralized CL, wherein each robot broadcasts its own information to the entire team. Then, acting as a Fusion Center (FC) [10], each robot can calculate and reproduce the centralized pose estimates. This scheme requires all-to-all robot communication at each information exchange and has a high-processing cost per robot. D-CL algorithms distributing computations of an EKF centralized CL algorithm is proposed in [3] and [11]. Subsequently, [6] presents a maximum-a-posteriori D-CL algorithm in which all the robots in the team calculate parts of the centralized CL. A D-CL approach equivalent to a centralized CL is proposed in [12]. In order to handle limited communication ranges and time-varying communication graphs, this method uses an information transfer scheme wherein each robot broadcasts all of its locally available information to every robot within its communication radius at each time-step. This information includes the past and present measurements, as well as past measurements previously received from other robots. The main drawback of this algorithm is its high communication cost, which may not be feasible in applications with limited communication bandwidth.

EKF based CL algorithms due to their recursive nature and relative ease of implementation has been studied extensively. However, the EKF, due to linearization approximation, is known to be inconsistent for highly nonlinear systems. An alternative recursive estimation filter, the UKF proposed in [15] is proven to work more consistently than the EKF for systems with nonlinear state and measurement models. However, for large systems, UKF is computationally more expensive than EKF. In this note, we consider a CL algorithm based on UKF for a team of mobile robots with communication and computation capabilities. In particular, we present a server-client paradigm to distribute the computational cost of this algorithm among team members. The highest computational cost of the UKF comes from calculating the collective covariance matrix, and its square root. The square root of covariance matrix is required to create the sigma points for UKF's statistical linearization. A comparison between different matrix square root calculation methods within a UKF application of GPS/INS sensor fusion was presented

The authors are with the Department of Mechanical and Aerospace Engineering, University of California Irvine, Irvine, CA 92697, USA, vdinh2,solmaz@uci.edu

in [16] which indicated that the Cholesky decomposition method was best suited for UKF applications. Our server-client based algorithm is constructed based on identifying an appropriate Cholesky decomposition algorithm which allows a coordinated computational task allocation among team members. In our algorithm, the server is responsible for storing the overall data generated by the UKF CL method, mainly, a matrix of the size of the collective covariance matrix of the team, the sigma points of the team and the relative measurement data. While, the clients are the robotic team members which based on a pre-specified coordinated computational task allocation, read, process and write back data at the server to cooperatively implement a centralized equivalent UKF CL algorithm.

II. Preliminaries

In this section, we introduce our notation, terminology, and the description of the mobile robotic group we study.

Let $\mathbb{R}$ and $\mathbb{Z}_{\geq 0}$ denote, respectively, the set of real and nonnegative integer numbers, $\mathbb{M}_n$ represent the set of real positive definite matrices of dimension $n \times n$, $\mathbf{0}_{n\times m}$ be the zero matrix of dimension $n \times m$, and $\mathbf{I}_n$ denote the identity matrix of dimension $n \times n$. The transpose of matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ is $\mathbf{A}^\top$. We denote the ith column of a matrix $\mathbf{A}$ by $[\mathbf{A}]_i$. For $n, m \in \mathbb{Z}_{\geq 0}$, with $n < m$, $n : m$ represents $\{n, \cdots, m\}$. For block partitioned matrix $\mathbf{A}$, $\mathbf{A}_{i:j,k:l}$ where $i < j$ and $k < l$, indicates a submatrix of $\mathbf{A}$ consisted of the blocks in the intersection of rows i to j and the columns k to l. For finite sets V_1 and V_2, we denote by $V_1 \backslash V_2$ the set whose elements consist of all the elements of V_1 that are not in V_2. We distinguish the variables associated to robot i by the superscript i, e.g., $\mathbf{x}^i$ is the pose (i.e., position and orientation) of robot i, $\hat{\mathbf{x}}^i$ is its pose estimate, and $\mathbf{P}^i$ is the covariance matrix of its pose estimate. We use $\mathbf{P}_{i,i}$ interchangeably with $\mathbf{P}^i$. The cross-covariance [1] of the pose vectors of robots i and j is $\mathbf{P}_{i,j}$. We denote the propagated and updated variables, say $\hat{\mathbf{x}}^i$, at time-step k by $\hat{\mathbf{x}}^{i-}(k)$ and $\hat{\mathbf{x}}^{i+}(k)$, respectively. We drop the time-step argument of the variables whenever it is clear from the context. If $\mathbf{q}^i \in \mathbb{R}^{n^i}$ is a local variable at robot i in a team of N robots, the aggregated $\mathbf{q}^i$'s is represented by $\mathbf{q} = (\mathbf{q}^1, \ldots, \mathbf{q}^N) \in \mathbb{R}^d$, $d = \sum_{i=1}^N n^i$.

A. Description of the mobile robot group

We consider a team of N mobile robots with processing and communication capabilities. We assume the communication links are reliable, i.e., information sent over the network will be received without errors at its destination. We also assume that there is a shared memory with a server (we refer to it in short as 'server') that robots can read or write data on it at every time the localization algorithm requires. Subsequently, when the robots are interacting with this server we refer to them as 'clients'. Every robot carries exteroceptive sensing devices to detect and take relative measurements (relative pose, range, bearing or a combination of them) from the team members in its measurement range. Every robot has a distinct detectable unique identification (UID) which, without loss of generality, here is represented by a unique number in $\mathcal{V} = \{1, \cdots, N\}$. The robotic team can be heterogeneous, and the motion of each robot is described by its own linear or nonlinear equations of motion

$$\mathbf{x}^i(k+1) = \mathbf{f}^i(\mathbf{x}^i(k), \mathbf{u}(k)) + \mathbf{B}^i(k)\boldsymbol{\eta}^i(k), \quad i \in \mathcal{V}.$$

Then, the collective motion equation of the team is given by

$$\mathbf{x}(k+1) = \mathbf{f}(\mathbf{x}(k), \mathbf{u}(k)) + \mathbf{B}(k)\boldsymbol{\eta}(k), \tag{1}$$

where $\mathbf{x}$, $\mathbf{u}$, and $\boldsymbol{\eta}$ are, respectively, the aggregated vectors of the pose $\mathbf{x}^i \in \mathbb{R}^{n^i}$, the input $\mathbf{u}^i \in \mathbb{R}^{m^i}$ and the process noise $\boldsymbol{\eta}^i \in \mathbb{R}^{n_{\mathrm{p}}^i}$, $i \in \mathcal{V}$. We use $n_x = \sum_{i=1}^N n^i$ to denote the size of the aggregated state of the mobile robotic team. Here, $\mathbf{f}(\mathbf{x}, \mathbf{u}) = (\mathbf{f}^1(\mathbf{x}^1, \mathbf{u}^1), \cdots, \mathbf{f}^N(\mathbf{x}^N, \mathbf{u}^N))$ and $\mathbf{B} = \mathrm{Diag}(\mathbf{B}^1, \cdots, \mathbf{B}^N)$, where, $\mathbf{f}^i(\mathbf{x}^i, \mathbf{u}^i)$ and $\mathbf{B}^i \in \mathbb{R}^{n^i \times n_{\mathrm{p}}^i}$, are, respectively, the system function and process noise coefficient matrix of the robot $i \in \mathcal{V}$. We assume that the process noises $\boldsymbol{\eta}^i \in \mathbb{R}^{n_{\mathrm{p}}^i}$, $i \in \mathcal{V}$, are independent zero-mean white Gaussian processes with a known positive definite variance $\mathbf{Q}^i = E[{\boldsymbol{\eta}^i}^\top \boldsymbol{\eta}^i]$. We model the relative measurement collected by robot i from robot j as

$$\mathbf{z}_{i,j}(k+1) = \mathbf{h}_{i,j}(\mathbf{x}^i(k), \mathbf{x}^j(k)) + \boldsymbol{\nu}^i(k), \quad \mathbf{z}_{i,j} \in \mathbb{R}^{n_z^i}, \tag{2}$$

where $\mathbf{h}_{i,j}(\mathbf{x}^i, \mathbf{x}^j)$ is the measurement model and $\boldsymbol{\nu}^i$ is the measurement noise of robot $i \in \mathcal{V}$, assumed to be independent zero-mean white Gaussian processes with known positive definite variance $\mathbf{R}^i = E[{\boldsymbol{\nu}^i}^\top \boldsymbol{\nu}^i]$. Here, we assume that all the sensor measurements are synchronized. Moreover, all sensor noises are assumed to be white and mutually uncorrelated. We show below how using a UKF, relative measurements between robots are used to improve the propagated states of the system.

B. Left-looking Cholesky decomposition over a server-client framework

In this section, we introduce the Cholesky decomposition algorithm which we use in our UKF algorithm. Let $\mathbf{P}$ be a real symmetric and positive definite matrix. Then there exists a real lower triangular matrix $\mathbf{L}$ so that $\mathbf{P} = \mathbf{L}\mathbf{L}^\top$ ($\mathbf{L}$ is unique if we restrict its diagonal elements to be positive). This is called the Cholesky decomposition. We refer to $\mathbf{L}$ as the Cholesky decomposition element. There are different techniques to obtain Cholesky decomposition (see e.g., [17]). In this paper, we use the *left-looking Cholesky decomposition* algorithm implemented on a two-level memory hierarchy (fast and slow), see Algorithm 1. In this algorithm, the matrix being factored initially resides in slow memory, and is too large to fit in the smaller fast memory. At every timestep, the fast memory works on a column of the decomposing matrix and uses the prior columns to obtain the respective Cholesky elements of that columns and writes back the results on

[1] We use the term *cross-covariance* to refer to the correlation terms between two robots in the covariance matrix of the entire team.

Algorithm 1 Left-looking Cholesky algorithm [17]

for $j = 1$ to n **do**
  read $P(j:n, j)$ from slow memory
  for $k = 1$ to $j-1$ **do**
    read $\mathbf{P}(j:n, k)$ from slow memory
    update diagonal element: $\mathbf{P}(j,j) \leftarrow \mathbf{P}(j,j) - \mathbf{P}(j,k)^2$
    for $i = j+1$ to n **do**
      update jth column element: $\mathbf{P}(i,j) \leftarrow \mathbf{P}(i,j) - \mathbf{P}(i,k)\mathbf{P}(j,k)$
    end for
  end for
  calculate final value of diagonal element $\mathbf{P}(j,j) \leftarrow \sqrt{\mathbf{P}(j,j)}$
  for $i = j+1$ to n **do**
    calculate final value of jth column element: $\mathbf{P}(i,j) \leftarrow \mathbf{P}(i,j)/\mathbf{P}(j,j)$
  end for
  write $\mathbf{P}(j:n, j)$ to slow memory
end for
$L = P$

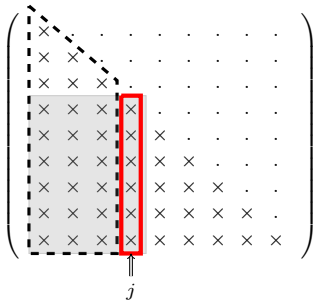

Fig. 1: Progression of the left-looking Cholesky algorithm over the matrix resided in the slow memory (server). j is the currently computed column at the fast memory (client). The dashed trapezoid is the part already computed. The gray part shows the matrix elements used in the computation of j.

the same column in the slow memory. Figure 1 depicts the progression of the left-looking Cholesky algorithm over the matrix resided in the slow memory. Starting with a matrix to decompose, the result of applying Algorithm 1 is to overwrite this matrix, on the slow memory, with its Cholesky decomposition element. The slow memory (server in our development below) is responsible to storing the matrix and the fast memory (client in out development below) is responsible to carrying out the calculations, a column at a time. The fact that this algorithm progresses one column at a time makes it an appropiriate candidate for our cooperative localization algorithm.

III. Benchmark Centralized Cooperative Localization Algorithm

In this section, we present a centralized UKF CL algorithm as our benchmark solution and evaluate its performance vs. and EKF CL algorithm in an example simulation. Our main contribution, presented in the next section, is to offer a novel sever-client based distribution of the computations of this algorithm among the members of the mobile robotic team.

Our centralized UKF CL algorithm below is the straightforward application of UKF (see [18]) over the collective model of the mobile robotic team described in Section II-A.

Centralized UKF CL Initialization

For $i \in \mathcal{V}$, we initialize the UKF algorithm at $\hat{\mathbf{x}}^{i+}(0) \in \mathbb{R}^{n^i}$, $\mathbf{P}^{i+}(0) \in \mathbb{M}_{n^i}$ and $\mathbf{P}^{+}_{i,j}(0) = \mathbf{0}_{n^i \times n^j}$, $j \in \mathcal{V} \backslash \{i\}$.

The prediction and correction steps of UKF over steps $k \in \mathbb{Z}_{\geq 0}$ is as follows.

Centralized UKF CL Prediction

At each prediction step, first, we obtain the Cholesky decomposition of the covariance matrix of the team, i.e., $\mathbf{P}^{+}(k) = \mathbf{L}(k)\mathbf{L}(k)^\top$ using Algorithm 1. Then, we use $\mathbf{L}(k)$ to construct $2n_x + 1$ sigma points of the UKF algorithm:

$$\boldsymbol{\chi}_{(0)} = \hat{\mathbf{x}}^{+}(k), \;\; w_{(0)} = \frac{\kappa}{(n_x + \kappa)} \tag{3a}$$

$$\boldsymbol{\chi}_{(l)} = \hat{\mathbf{x}}^{+}(k) + \sqrt{n_x + \kappa}[\mathbf{L}(k)]_l, \;\; w_{(l)} = \frac{1}{2(n_x + \kappa)} \tag{3b}$$

$$\boldsymbol{\chi}_{(l+n_x)} = \hat{\mathbf{x}}^{+}(k) - \sqrt{n_x + \kappa}[\mathbf{L}(k)]_l, \;\; w_{(l+n_x)} = \frac{1}{2(n_x + \kappa)}, \tag{3c}$$

where $l \in \{1, \cdots, n_x\}$. Here, κ is a design parameter in the selection of the sigma points, usually chosen so that $n_x + \kappa = 3$. This set of sigma points captures the moments of the underlying distribution up to the third order for the Gaussian case. Next, we obtain the transformed sigma points

$$\boldsymbol{\chi}^{-}_{(l)} = \mathbf{f}(\boldsymbol{\chi}_{(l)}(k), \mathbf{u}(k)), \quad l \in \{0, \cdots, 2n_x\}. \tag{4}$$

Then, the collective predicted UKF state and covariance equations are, respectively,

$$\hat{\mathbf{x}}^{-}(k+1) = \sum_{l=0}^{2n_x} w_{(l)} \, \boldsymbol{\chi}^{-}_{(l)}, \tag{5a}$$

$$\mathbf{P}^{-}(k+1) = \sum_{l=0}^{2n_x} w_{(l)} \, \mathbf{e}_{x,(l)} \mathbf{e}^\top_{x,(l)} + \mathbf{B}(k)\mathbf{Q}(k)\mathbf{B}(k)^\top, \tag{5b}$$

where $\mathbf{e}_{x,(l)} = \boldsymbol{\chi}^{-}_{(l)} - \hat{\mathbf{x}}^{-}(k+1)$, $l \in \{0, \cdots, 2n_x\}$.

Centralized UKF CL Correction

While there are no relative measurements in the team, no correction happens, therefore,

$$\hat{\mathbf{x}}^{+}(k+1) = \hat{\mathbf{x}}^{-}(k+1), \quad \mathbf{P}^{+}(k+1) = \mathbf{P}^{-}(k+1).$$

For simplicity, we assume that only one relative measurement takes place at each time. Multiple relative measurements can be handled by sequential updating procedure (c.f. [19], [20]). Let robot a make a relative measurement from robot b. The UKF correction equation is obtained as follows. First, we calculate the measurement model using predicted sigma points. In relative measurement models, we only need the sigma points corresponding to robot a and b, i.e.,

$$\boldsymbol{\zeta}_{ab,(l)} = \mathbf{h}_{ab}(\boldsymbol{\chi}^{a-}_{(l)}, \boldsymbol{\chi}^{b-}_{(l)}), \quad l \in \{0, \cdots, 2n_x\}. \tag{6}$$

Then, the predicted relative measurement, the measurement residual and the innovation covariance are, respectively,

$$\hat{\mathbf{z}}_{ab} = \sum_{l=0}^{2n_x} w_{(l)} \boldsymbol{\zeta}_{ab,(l)}, \tag{7a}$$

$$\mathbf{r}^a = \mathbf{z}_{ab} - \hat{\mathbf{z}}_{ab}, \tag{7b}$$

$$\mathbf{S}_{ab} = \sum_{l=0}^{2n_x} w_{(l)} \, \mathbf{e}_{z,(l)} \mathbf{e}^\top_{z,(l)} + \mathbf{R}^a(k). \tag{7c}$$

where $\mathbf{e}_{z,(l)} = \boldsymbol{\zeta}_{ab,(l)} - \hat{\mathbf{z}}_{ab}$. Let

$$\mathbf{P}_{\text{XZ}} = \sum_{l=0}^{2n_x} w_{(l)} \, \mathbf{e}_{x,(l)} \mathbf{e}^\top_{z,(l)}.$$

Then, the UKF gain is given by

$$\mathbf{K}(k+1) = \mathbf{P}_{\text{XZ}} \, {\mathbf{S}_{ab}}^{-1}. \tag{8}$$

Finally, the corrected collective team estimations are

$$\hat{\mathbf{x}}^{+}(k+1) = \hat{\mathbf{x}}^{-}(k+1) + \mathbf{K}(k+1)\mathbf{r}^a, \tag{9a}$$

$$\mathbf{P}^{+}(k+1) = \mathbf{P}^{-}(k+1) - \mathbf{K}(k+1)\mathbf{S}_{ab}\mathbf{K}(k+1)^\top, \tag{9b}$$

TABLE I: Computational cost per time-step of the centralized UKF CL algorithm of Section III in terms of n_x, the size of the collective state of the robotic team.

prediction stage, total cost $O(n_x^3)$				
$\mathbf{L}(k)$ (cf. [17])	(3)	(4)	(5a)	(5b)
$O(n_x^3)$	$O(n_x)$	$O(n_x)$	$O(n_x)$	$O(n_x^2)$
correction stage, total cost $O(n_x^2)$				
(7a)	(7c)	(8)	(9a)	(9b)
$O(n_x)$	$O(n_x^2)$	$O(n_x^2)$	$O(n_x)$	$O(n_x^2)$

Table I presents the computational cost of the aforementioned centralized UKF CL algorithm at each time step in terms of n_x, the size of the collective state of the team. Figure 2 demonstrates a comparative performance study of UKF-based and EKF-based CL algorithms.

IV. Computationally distributed server-client based UKF for cooperative localization

In this section, we introduce our server-client based algorithm for UKF CL which intends to distribute the high computational cost of a centralized operation among the robotic team members. We start by expanding the collective UKF CL equations of Section III in terms of robot-wise components of the aggregated state and covariance equations of the team. While doing this, we make observations about the nature of the source of coupling in the equations of robots and use these observations to, systematically, generate our server-client based algorithm for UKF CL.

We partition the the Cholesky decomposition element $\mathbf{L}$ of the state covariance matrix $\mathbf{P}^{+}(k)$ in a comparable way to its robot-wise partitions, i.e., $\mathbf{L}_{i,i}(k) \in \mathbb{R}^{n^i \times n^i}$ and $\mathbf{L}_{i,j}(k) \in \mathbb{R}^{n^i \times n^j}$, for $i \in \mathcal{V}$ and $j \in \mathcal{V}\backslash\{i\}$, where $\mathbf{L}_{i,i}$'s are lower triangular matrices. An example of such partitioning is given below for a group of $N = 3$ robots:

$$\underbrace{\left[\begin{array}{c|c|c} \mathbf{P}^{+}_{1,1}(k) & \mathbf{P}^{+}_{1,2}(k) & \mathbf{P}^{+}_{1,3}(k) \\ \hline \mathbf{P}^{+}_{2,1}(k) & \mathbf{P}^{+}_{2,2}(k) & \mathbf{P}^{+}_{2,3}(k) \\ \hline \mathbf{P}^{+}_{3,1}(k) & \mathbf{P}^{+}_{3,2}(k) & \mathbf{P}^{+}_{3,3}(k) \end{array}\right]}_{\mathbf{P}^{+}(k)}, \quad \underbrace{\left[\begin{array}{c|c|c} \ddots \quad \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{L}_{1,1} \quad \ddots & & \\ \hline \mathbf{L}_{2,1} & \mathbf{L}_{2,2} \ddots & \mathbf{0} \\ \hline \mathbf{L}_{3,1} & \mathbf{L}_{3,2} & \mathbf{L}_{3,3} \ddots \end{array}\right]}_{\mathbf{L}(k)},$$

By taking into account the structure of $\mathbf{L}(k)$, the sigma points (3) can be expanded in terms of robot-wise components as follows

$$\boldsymbol{\chi}^i_{(l)} = \hat{\mathbf{x}}^{i+}(k), \quad l \in \{0, \cdots n_x\}\backslash\{1, \cdots, \sum_{j=1}^{i} n^j\}, \quad \text{(10a)}$$

$$\boldsymbol{\chi}^i_{(l+n_x)} = \hat{\mathbf{x}}^{i+}(k), \quad l \in \{1, \cdots n_x\}\backslash\{1, \cdots, \sum_{j=1}^{i} n^j\}, \quad \text{(10b)}$$

$$\boldsymbol{\chi}^i_{(l)} = \hat{\mathbf{x}}^{i+}(k) + \sqrt{n_x+\kappa}[\mathbf{L}_{i,1:i}]_l, \quad l \in \{1, \cdots, \sum_{j=1}^{i} n^j\}, \quad \text{(10c)}$$

$$\boldsymbol{\chi}^i_{(l+n_x)} = \hat{\mathbf{x}}^{i+}(k) - \sqrt{n_x+\kappa}[\mathbf{L}_{i,1:i}]_l, \quad l \in \{1, \cdots, \sum_{j=1}^{i} n^j\}. \quad \text{(10d)}$$

Observation 1: Observe that in (10) the sigma points of each robot depend only on its own predicted state and the elements of corresponding horizontal partition of matrix $\mathbf{L}$.

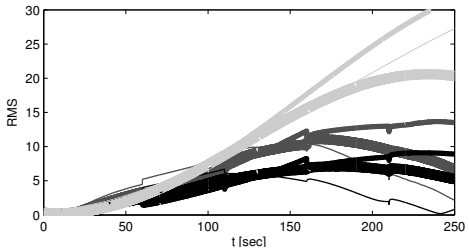

Fig. 2: Position root mean square error (RMS) of 30 Monte Carlo simulations for 3 robots moving on a flat terrain ($\mathbf{x}^i = [x, y, \phi]^\top$). The relative measurement scenario, used in all simulations, consists of robot 3 taking alternating relative pose measurement over consecutive 50 seconds time intervals from robot 1 and 2, starting at $t = 10$ seconds. The very thin curves correspond to robot 1, the thick curves correspond to robot 2 and the very thick curves correspond to robot 3. The black, gray and light gray curves correspond to, respectively, the UKF-based CL, the EKF-based CL, and UKF localization without CL (every robot propagates its model). As this figure show, employing both UKF-based and EKF-based CL improves robot localizations. However, the UKF strategy results in further improvement due to its less conservative method to handle system nonlinearities.

For example, for a team of three robots with $n^1 = n^2 = n^3 = 2$, some of the sigma points for the robots are

$$\boldsymbol{\chi}^1_{(1)} = \hat{\mathbf{x}}^{1+}(k) + \sqrt{n_x+\kappa}[\mathbf{L}_{1,1}]_1,$$
$$\boldsymbol{\chi}^2_{(1)} = \hat{\mathbf{x}}^{2+}(k) + \sqrt{n_x+\kappa}[\mathbf{L}_{2,1} \quad \mathbf{L}_{2,2}]_1,$$
$$\boldsymbol{\chi}^3_{(1)} = \hat{\mathbf{x}}^{3+}(k) + \sqrt{n_x+\kappa}[\mathbf{L}_{3,1} \quad \mathbf{L}_{3,2} \quad \mathbf{L}_{3,3}]_1,$$

where $\mathbf{L}_{1,1} \in \mathbb{R}^{2\times 2}$, and $[\mathbf{L}_{2,1} \quad \mathbf{L}_{2,2}] \in \mathbb{R}^{2\times 4}$, and $[\mathbf{L}_{3,1} \quad \mathbf{L}_{3,2} \quad \mathbf{L}_{3,3}] \in \mathbb{R}^{2\times 6}$.

Because the equation of each robot is decoupled from the rest of the team, the transformed sigma points of each robot $i \in \mathcal{V}$, only depend on the local equations of the robot, i.e.,

$$\boldsymbol{\chi}^{i-}_{(l)} = \mathbf{f}^i(\boldsymbol{\chi}^i_{(l)}(k), \mathbf{u}^i(k)), \quad l \in \{0, \cdots, 2n_x\}. \quad \text{(11)}$$

Then, the predicted UKF state and covariance equations of each robot $i \in \mathcal{V}$ are, respectively,

$$\hat{\mathbf{x}}^{i-}(k+1) = \sum_{l=0}^{2n_x} w_{(l)}\, \boldsymbol{\chi}^{i-}_{(l)}, \quad \text{(12a)}$$

$$\mathbf{P}^{-}_{i,i}(k+1) = \sum_{l=0}^{2n_x} w_{(l)}\, \mathbf{e}^i_{x,(l)} (\mathbf{e}^i_{x,(l)})^\top + \mathbf{B}^i(k)\mathbf{Q}^i(k)\mathbf{B}^i(k)^\top, \quad \text{(12b)}$$

The cross-covariance terms among the robots are given by

$$\mathbf{P}^{-}_{i,j}(k+1) = \sum_{l=0}^{2n_x} w_{(l)}\, \mathbf{e}^i_{x,(l)} (\mathbf{e}^j_{x,(l)})^\top, \quad \text{(13)}$$

for $i \in \mathcal{V}$ and $j \in \mathcal{V}\backslash\{i\}$.

Observation 2: Once the sigma points are obtained, the predicted state and state error covariance of each robot can be obtained locally at each robot. The source of coupling between robots is the error cross-covariance terms among the team members. Therefore, robots need to collaborate with one another to calculate these terms.

Next, notice that, we can obtain measurement sigma points (6) for $l \in \{0, \cdots, 2n_x\}$ as

$$\boldsymbol{\zeta}_{ab,(l)} = \mathbf{h}_{ab}(\mathbf{e}^a_{x,(l)} + \hat{\mathbf{x}}^{a-}, \mathbf{e}^b_{x,(l)} + \hat{\mathbf{x}}^{b-}), \quad \text{(14)}$$

where we used $\boldsymbol{\chi}^{j\text{-}}_{(l)} = \mathbf{e}^{j}_{x,(l)} + \hat{\mathbf{x}}^{j\text{-}}$, $j \in \{a, b\}$.

We partition $\mathbf{K}$ as $\mathbf{K} = \left[\mathbf{K}_1^\top, \cdots, \mathbf{K}_N^\top\right]^\top$. Here, $\mathbf{K}_i \in \mathbb{R}^{n^i \times n_z^a}$ is the portion of the UKF gain used to correct the pose estimate of the robot $i \in \mathcal{V}$, and is given by

$$\mathbf{K}_i = \sum\nolimits_{l=0}^{2n_x} w_{(l)}\, \mathbf{e}^{i}_{x,(l)} (\mathbf{e}_{z,(l)})^\top \mathbf{S}_{ab}{}^{-1}. \tag{15}$$

Then, the corrected pose and covariance equations (9) can be represented as follows where $i \in \mathcal{V}$ and $j \in \mathcal{V}\backslash\{i\}$

$$\hat{\mathbf{x}}^{i\text{+}}(k{+}1) = \hat{\mathbf{x}}^{i\text{-}}(k{+}1) + \mathbf{K}_i \mathbf{r}^a, \tag{16a}$$

$$\mathbf{P}^{+}_{i,i}(k{+}1) = \sum\nolimits_{l=0}^{2n_x} w_{(l)} \mathbf{e}^{i}_{x,(l)} \mathbf{e}^{i}_{x,(l)}{}^\top - \mathbf{K}_i \mathbf{S}_{ab}^{-1} \mathbf{K}_i^\top, \tag{16b}$$

$$\mathbf{P}^{+}_{i,j}(k{+}1) = \sum\nolimits_{l=0}^{2n_x} w_{(l)} \mathbf{e}^{i}_{x,(l)} \mathbf{e}^{j}_{x,(l)}{}^\top - \mathbf{K}_i \mathbf{S}_{ab}^{-1} \mathbf{K}_j^\top. \tag{16c}$$

Observation 3: After obtaining the measurement information, each robot can calculate its corresponding UKF gains and process its corrected state and state covariance locally. The source of coupling between robots is the state cross-covariance terms among the team members.

Based on the observations above, we propose a server-client based approach to distribute the computational load of the aforementioned UKF CL among the robotic team members. We assume that there is a shared memory that is accessible by every robot/client with a server that lets clients read and write data on this shared memory. The shared memory allocates the memory for the following variables (see Fig. 3):

- $\mathbf{M}$ where $\mathbf{e}_{x,(l)}$, $l \in \{0, \cdots, 2n_x\}$ of the entire team will reside at every timestep;
- $\bar{\mathbf{M}}$ where at each timestep, first $\mathbf{P}^{+}$ will reside in it and next its Cholesky decomposition matrix element $\mathbf{L}$. Because $\mathbf{P}^{+}$ is symmetric and $\mathbf{L}$ is lower triangular, we only need to allocate memory to the lower block triangular portion as shown in Fig. 3;
- $\tilde{\mathbf{M}}$ where, respectively, the identity of the robot taking the relative measurement (i.e., a), the predicted state of this robot a, the measurement vector ($\mathbf{z}_{ab}$), the identity of the robot the measurement is taken from it (i.e, b), and finally the predicted state of this robot b will reside in this variable;
- $\bar{\mathbf{M}}$ where the UKF gain of the robots will reside.

Initialization

All the initial cross-covariance terms between robots are equal to zero. Every robot/client $i \in \mathcal{V}$ initializes its estimation variables $\hat{\mathbf{x}}^{i\text{+}}(0) \in \mathbb{R}^{n^i}$, and $\mathbf{P}^{+}_{i,i}(0) \in \mathbb{M}_{n^i}$. Then, every robot/client writes its initial $\mathbf{P}^{+}_{i,i}(0)$ at the corresponding location of $\bar{\mathbf{M}}$ in the shared memory. Notice that because the initial cross-covariance terms are equal to zero, the off-diagonal blocks of $\bar{\mathbf{M}}$ are all zero, i.e, at $k = 0$, $\bar{\mathbf{M}}_{i,i} = \mathbf{P}^{+}_{i,i}(0)$, $\bar{\mathbf{M}}_{i,j} = \mathbf{0}$, $i \in \mathcal{V}$, $j \in \mathcal{V}\backslash\{i\}$.

The prediction and correction steps and the interaction between the server and the clients of our proposed computationally distributed UKF over steps $k \in \mathbb{Z}_{\geq 0}$ are as follows.

Prediction stage

The first step in the prediction stage is to calculate the Cholesky decomposition and to compute the sigma points of each robot. To accomplish this task, given the facts that a) the left-looking Cholesky decomposition progresses one column at a time and b) to compute its sigma points (10) every robot $i \in \mathcal{V}$ only needs $\mathbf{L}_{i:N,1:i}$ partitions of the Cholesky decomposition element $\mathbf{L}$, we propose the following cooperative compute sequential procedure. Robot/client $i \in \mathcal{V}$ reads the blocks $\bar{\mathbf{M}}_{i:N,1:i}$ from the server, and preforms the corresponding steps of the Algorithm 1's Cholesky decomposition over blocks $\bar{\mathbf{M}}_{i:N,i}$ to obtain $\mathbf{L}_{i:N,i}$. Once finished, robot i writes back $\mathbf{L}_{i:N,i}$ over $\bar{\mathbf{M}}_{i:N,i}$ on the server. Then, the next robot starts the same procedure. Eventually, robot N will compute the $\mathbf{L}_{N,N}$. Notice that every robot/client $i \in \mathcal{V}$ is reading $\bar{\mathbf{M}}_{i:N,1:i} = [\mathbf{L}_{i:N,1:i-1} \;\; \bar{\mathbf{M}}_{i:N,i}]$ and computing $\mathbf{L}_{i:N,i}$ and eventually over writing $\bar{\mathbf{M}}_{i:N,i}$ with its computed $\mathbf{L}_{i:N,i}$. Once each robot/client $i \in \mathcal{V}$ is done with calculating its corresponding $\mathbf{L}_{i:N,i}$, it computes its own sigma points (10) locally. Notice that following the decomposition procedure above, every robot $i \in \mathcal{V}$, has access to the corresponding row block of the Cholesky decomposition element $\mathbf{L}(k)$ needed to compute its local sigma points (10).

Next, every robot $i \in \mathcal{V}$, computes locally its transformed sigma points (11), and subsequently computes its local predicted state (12a) and the corresponding error covariance (12b). Then every robot $i \in \mathcal{V}$ writes its $\mathbf{e}^{i}_{x,(l)}$, $l \in \{0, \cdots, 2n_x\}$, on the corresponding row block of $\mathbf{M}$ at the server. Once all robots are done, we have $\mathbf{M} = (\mathbf{e}_{x,(0)}, \mathbf{e}_{x,(1)}, \cdots, \mathbf{e}_{x,(2n_x)})$.

Correction stage

When there is a relative measurement in the team, the robot that has taken the measurement, denoted by robot a, writes its identity at $\tilde{\mathbf{M}}_{1,1}$, its current predicted state $\mathbf{x}^{a\text{-}}$ at $\tilde{\mathbf{M}}_{2,1}$, its measurement vector at $\tilde{\mathbf{M}}_{3,1}$ and the UID of the landmark robot b at $\tilde{\mathbf{M}}_{4,1}$. Then, every robot $i \in \mathcal{V}\backslash\{a\}$ reads $\tilde{\mathbf{M}}_{1:4,1}$, from the shared memory. Subsequently, robot b writes its current predicted state $\mathbf{x}^{b\text{-}}$ in $\tilde{\mathbf{M}}_{5,1}$. Then every robot reads $\tilde{\mathbf{M}}_{5,1}$ from the shared memory.

We assume that every robot knows the measurement model of all the other robots in the team. Then, by reading $\tilde{\mathbf{M}}$ and the rows of $\mathbf{M}$ corresponding to robots $\{a, b\}$ from the shared memory, every robot can calculate a local copy of the measurement variables (7) (recall (14)). Subsequently, every robot $i \in \mathcal{V}$ can calculate its corresponding UKF gain $\mathbf{K}_i$ given (15), and as a result its corrected state (16a) and error covariance (16b) locally by itself.

Next, robots should create the cross-covariance matrices (16a) and write the corrected covariance matrix of the team $\mathbf{P}^{+}$ on the share memory for next step. To accomplish this task, the robots follow the parallel procedure in Procedure 1 below (each robot i can perform this stage in parallel with other robots). As final task for the correction stage, robot a writes zero at both $\tilde{\mathbf{M}}_{1,1}$ and $\tilde{\mathbf{M}}_{4,1}$.

If there is no relative measurement at the current time-step

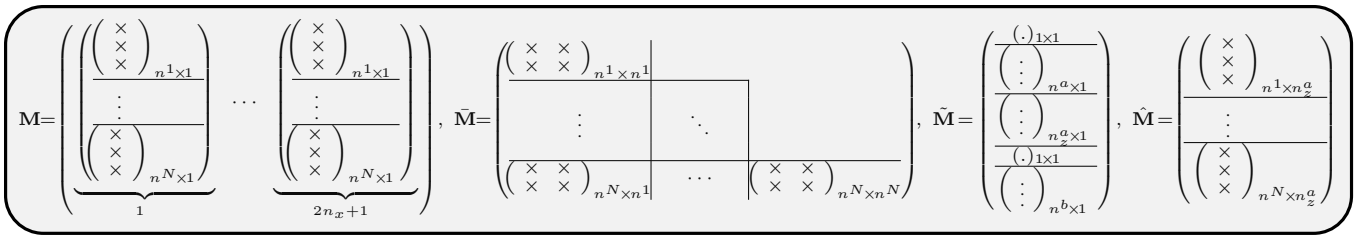

Fig. 3: Matrices stored in the server/shared memory. The blank blocks of $\bar{\mathbf{M}}$ are not needed and no memory is allocated for them.

Procedure 1: distributed calculation of team cross-covariances

1: **for** robot $i = 1$ to N **do**
2: read $\hat{\mathbf{M}}_{(i:N,1)}$ from the shared memory to obtain $\mathbf{K}_{i:N}$
3: **for** $l = i - 1$ to $2n_x$ **do**
4: read $\mathbf{M}(i : N, l + 1)$ from shared memory to obtain $\mathbf{e}^{i:N}_{x,(l)}$
5: **for** $j = i$ to N **do**
6: set $\mathbf{P}^{+}_{i,j} \leftarrow \mathbf{P}^{+}_{i,j} + \mathbf{e}^{i}_{x,(l)}(\mathbf{e}^{j}_{x,(l)})^{\top}$
7: **end for**
8: **end for**
9: **for** $j = i$ to N **do**
10: $\mathbf{P}^{+}_{i,j} \leftarrow \mathbf{P}^{+}_{i,j} - \mathbf{K}_i \mathbf{S}^{-1}_{ab} \mathbf{K}^{\top}_j$
11: write $\mathbf{P}^{+}_{i,j}$ in the corresponding block at $\bar{\mathbf{M}}_{i,j}$ in the shared memory
12: **end for**
13: **end for**

$k + 1$, every robot $i \in \mathcal{V}$ obtains its estimation as

$$\hat{\mathbf{x}}^{i+}(k+1) = \hat{\mathbf{x}}^{i-}(k+1), \quad \mathbf{P}^{+}_{i,i}(k+1) = \mathbf{P}^{-}_{i,i}(k+1).$$

Then, the robots need to proceed with the distributed calculation of the team cross-covariance terms as outlined in Procedure 1 expect for steps 2 and 10. When there is no relative measurement at the current time-step, the robots do not need to read from or write on $\tilde{\mathbf{M}}$ and $\hat{\mathbf{M}}$ variables of the shared memory except for $\tilde{\mathbf{M}}_{1,1}$ which its zero value indicates there is no relative measurement in the team.

It worth noticing that except for the Cholesky decomposition, which must be done in a sequential manner, the rest of calculations by each robot is independent from the rest of the team and can be done in a parallel manner.

V. Conclusions

For a team of robots with communication and computational capabilities, we presented a cooperative localization algorithm based on a UKF whose computations are distributed among the team members via a server-client paradaim. Our server-client based computationally distributed algorithm is based on identifying an appropriate Cholesky decomposition algorithm which allows a coordinated computational task allocation among team member.

References

[1] J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 376–382, June 1991.

[2] G. Dissanayake, P. Newman, H. F. Durrant-Whyte, , S. Clark, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001.

[3] S. I. Roumeliotis and G. A. Bekey, "Distributed multirobot localization," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 781–795, 2002.

[4] S. Panzieri, F. Pascucci, and R. Setol, "Multirobot localisation using interlaced extended kalman filter," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pp. 2816–2821, Oct. 2006.

[5] N. Karam, F. Chausse, R. Aufrere, and R. Chapuis, "Localization of a group of communicating vehicles by state exchange," in *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pp. 519–524, Oct. 2006.

[6] E. D. Nerurkar, S. I. Roumeliotis, and A. Martinelli, "Distributed maximum a posteriori estimation for multi-robot cooperative localization," in *IEEE Int. Conf. on Robotics and Automation*, (Kobe, Japan), pp. 1402–1409, May 2009.

[7] S. I. Roumeliotis, *Robust mobile robot localization: from single-robot uncertainties to multi-robot interdependencies*. PhD thesis, University of Southern California, 2000.

[8] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed scalable solution to the area coverage problem," in *Int. Conference on Distributed Autonomous Robotic Systems*, (Fukuoka, Japan), pp. 299–308, June 2002.

[9] A. Bahr, M. R. Walter, and J. J. Leonard, "Consistent cooperative localization," in *IEEE Int. Conf. on Robotics and Automation*, (Kobe, Japan), pp. 8908–8913, May 2009.

[10] N. Trawny, S. I. Roumeliotis, and G. B. Giannakis, "Cooperative multi-robot localization under communication constraints," in *IEEE Int. Conf. on Robotics and Automation*, (Kobe, Japan), pp. 4394–4400, May 2009.

[11] S. S. Kia, S. Rounds, and S. Martínez, "A centralized-equivalent decentralized implementation of extended Kalman filters for cooperative localization," in *IEEE/RSJ Int. Conference on Intelligent Robots and Systems*, (Chicago, IL), pp. 3761–3765, September 2014.

[12] K. Y. K. Leung, T. D. Barfoot, and H. H. T. Liu, "Decentralized localization of sparsely-communicating robot networks: A centralized-equivalent approach," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 62–77, 2010.

[13] A. T. Ihler, J. W. Fisher, R. L. Moses, and A. S. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE Journal of Selected Areas in Communications*, vol. 23, no. 4, pp. 809–819, 2005.

[14] J. Nilsson, D. Zachariah, I. Skog, and P. Händel, "Cooperative localization by dual foot-mounted inertial sensors and inter-agent ranging," *EURASIP Journal on Advances in Signal Processing*, vol. 2013, no. 164, 2013.

[15] S. J. Julier and J. K. Uhlmann, "A new extension of the kalman filter to nonlinear systems," in *AeroSense: the 11th International Symposium on Aerospace/Defence Sensing, Simulation and Controls*, pp. 182–193, 1997.

[16] M. Rhudy, Y. Gu, J. Gross, and M. R. Napolitano, "Evaluation of matrix square root operations for UKF within a UAV-based GPS/INS sensor fusion applicationn," *International Journal of Navigation and Observation*, vol. 2011, no. 1, pp. 1–11, 2011.

[17] G. Ballard, J. Demmel, O. Holtz, and O. Schwartz, "Communication-optimal parallel and sequential cholesky decomposition," *SIAM Journal on Scientific Computing*, vol. 32, no. 6, pp. 3495–3523, 2010.

[18] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[19] C. T. Leondes, ed., *Advances in Control Systems Theory and Application*, vol. 3. New York: Academic Press, 1966.

[20] Y. Bar-Shalom, P. K. Willett, and X. Tian, *Tracking and Data Fusion, a Handbook of Algorithms*. Storts, CT, USA: YBS Publishing, 2011.