# Distributed Maximum Likelihood for Simultaneous Self-Localization and Tracking in Sensor Networks

Nikolas Kantas, Sumeetpal S. Singh, and Arnaud Doucet

***Abstract*—We show that the sensor self-localization problem can be cast as a static parameter estimation problem for Hidden Markov Models and we implement fully decentralized versions of the Recursive Maximum Likelihood and on-line Expectation-Maximization algorithms to localize the sensor network simultaneously with target tracking. For linear Gaussian models, our algorithms can be implemented exactly using a distributed version of the Kalman filter and a novel message passing algorithm. The latter allows each node to compute the local derivatives of the likelihood or the sufficient statistics needed for Expectation-Maximization. In the non-linear case, a solution based on local linearization in the spirit of the Extended Kalman Filter is proposed. In numerical examples we demonstrate that the developed algorithms are able to learn the localization parameters.**

***Index Terms*—Collaborative tracking, sensor localization, target tracking, maximum likelihood, sensor networks.**

## I. Introduction

THIS paper is concerned with sensor networks that are deployed to perform target tracking. A network is comprised of synchronous sensor-trackers where each node in the network has the processing ability to perform the computations needed for target tracking. A moving target will be simultaneously observed by more than one sensor. If the target is within the field-of-view of a sensor, then that sensor will collect measurements of the target. Traditionally in tracking a centralized architecture is used whereby all the sensors transmit their measurements to a central fusion node, which then combines them and computes the estimate of the target's trajectory. However, here we are interested in performing *collaborative tracking*, but without the need for a central fusion node. Loosely speaking, we are interested in developing distributed tracking algorithms for networks whose nodes collaborate by exchanging appropriate messages between neighboring nodes to achieve the same effect as they would by communicating with a central fusion node.

Manuscript received March 27, 2012; accepted May 24, 2012. Date of publication June 25, 2012; date of current version September 11, 2012. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Maciej Niedzwiecki. The work of N. Kantas was supported by the Engineering and Physical Sciences Research Council programme grant on Control For Energy and Sustainability (EP/G066477/1). The work of S. S. Singh was supported in part by the Engineering and Physical Sciences Research Council under the First Grant Scheme (EP/G037590/1).

N. Kantas is with the Control and Power Group, Department of Electrical and Electronic Engineering, Imperial College, London, U.K., SW7 2AZ (e-mail: n.kantas@imperial.ac.uk).

S. S. Singh is with the Signal Processing Laboratory, Department of Engineering, University of Cambridge, Cambridge, U.K., CB2 1PZ (e-mail: sss40@cam.ac.uk).

A. Doucet is with the Department of Statistics, University of Oxford, Oxford, U.K. OX1 3TG (e-mail: doucet@stats.ox.ac.uk).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2012.2205923

A necessary condition for distributed collaborative tracking is that each node is able to accurately determine the position of its neighboring nodes in its local frame of reference. (More details in Section II.) This is essentially an instance of the *self-localization* problem. In this work we solve the self-localization problem in an on-line manner. By on-line we mean that self-localization is performed on-the-fly as the nodes collect measurements of the moving target. In addition, given the absence of a central fusion node collaborative tracking and self-localization have to be performed in a fully *decentralized* manner, which makes necessary the use of message passing between neighboring nodes.

There is a sizable literature on the self-localization problem. The topic has been independently pursued by researchers working in different application areas, most notably wireless communications [1]–[5]. Although all these works tend to be targeted for the application at hand and differ in implementation specifics, they may however be broadly summarized into two categories. Firstly, there are works that rely on direct measurements of distances between neighboring nodes [2]–[5]. The latter is usually estimated from the Received Signal Strength (RSS) when each node is equipped with a wireless transceiver. Given such measurements, it is then possible to solve for the geometry of the sensor network but with ambiguities in translation and rotation of the entire network remaining. These ambiguities can be removed if the absolute position of certain nodes, referred to as anchor nodes, are known. Another approach to self-localization utilizes *beacon* nodes which have either been manually placed at precise locations, or their locations are known using a Global Positioning System (GPS). The un-localized nodes will use the signal broadcast by these beacon nodes to self-localize [1], [6]–[8]. We emphasize that in the aforementioned papers self-localization is performed off-line. The exception is [8], where they authors use Maximum Likelihood (ML) and Sequential Monte Carlo (SMC) in a centralized manner.

In this paper we aim to solve the localization problem without the need of a GPS or direct measurements of the distance between neighboring nodes. The method we propose is significantly different. Initially, the nodes do not know the relative locations of other nodes, so they can only behave as independent trackers. As the tracking task is performed on objects that traverse the field of view of the sensors, information is shared between nodes in a way that allows them to self-localize. Even though the target's true trajectory is not known to the sensors, localization can be achieved in this manner because the same target is being simultaneously measured by the sensors. This simple fact, which with the exception of [9]–[11] seems to have been overlooked in the localization literature, is the basis of our solution. However, our work differs from [9], [10] in the application studied as well as the inference scheme. Both [9], [10] formulate the localization as a Bayesian inference problem and approximate the posterior distributions of interest with Gaussians. Reference [10] uses a moment matching method and appears to be centralized in nature. The method in [9] uses instead linearization, is distributed and on-line, but its implementation relies on communication via a junction tree (see [12] for details) and requires an anchor node as pointed out in ([13, Sect. 6.2.3]). In this paper we formulate the sensor localization problem as a static parameter estimation problem for Hidden Markov Models (HMMs) [14], [15] and we estimate these static parameters using a ML approach, which has not been previously developed for the self-localization problem. We implement fully *decentralized* versions of the two most common on-line ML inference techniques, namely Recursive Maximum Likelihood (RML) [16]–[18] and on-line Expectation-Maximization (EM) [19]–[21]. A clear advantage of this approach compared to previous alternatives is that it makes an on-line implementation feasible. Finally, [11] is based on the principle shared by our approach and [9], [10]. In [11] the authors exploit the correlation of the measurements made by the various sensors of a hidden spatial process to perform self-localization. However for reasons concerned with the applications being addressed, which is not distributed target tracking, their method is not on-line and is centralized in nature.

The structure of the paper is as follows. We begin with the specification of the statistical model for the localization and tracking problem in Section II. In Section III we show how message passing may be utilized to perform distributed filtering. In Section IV we derive the distributed RML and on-line EM algorithms. Section V presents several numerical examples on small and medium sized networks. In Sections VI we provide a discussion and a few concluding remarks. The Appendix contains more detailed derivations of the distributed versions of RML and EM.

## II. Problem Formulation

We consider the sensor network $(\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ denotes the set of nodes of the network and $\mathcal{E}$ is the set of edges (or communication links between nodes). We will assume that the sensor network is connected, i.e., for any pair of nodes $i, j \in \mathcal{V}$ there is at least one path from $i$ to $j$. Nodes $i, j \in \mathcal{V}$ are adjacent or neighbors provided the edge $(i, j) \in \mathcal{E}$ exists. Also, we will assume that if $(i, j) \in \mathcal{E}$, then $(j, i) \in \mathcal{E}$ as well. This implies is that communication between nodes is bidirectional. The nodes observe the same physical target at discrete time intervals $n \in \mathbb{N}$. We will assume that all sensor-trackers are synchronized with a common clock and that the edges joining the different nodes in the network correspond to reliable communication links. These links define a neighborhood structure for each node and we will also assume that each sensor can only communicate with its neighboring nodes.

The hidden state, as is standard in target tracking, is defined to comprise of the position and velocity of the target, $X_n^r = [X_n^r(1), X_n^r(2), X_n^r(3), X_n^r(4)]^{\mathrm{T}}$, where $X_n^r(1)$ and $X_n^r(3)$ is the target's $x$ and $y$ position while $X_n^r(2)$ and $X_n^r(4)$ is the velocity in the $x$ and $y$ direction. Subscript $n$ denotes time while superscript $r$ denotes the coordinate system w.r.t. which these quantities are defined. For generality we assume that each node maintains a local coordinate system (or frame of reference) and regards itself as the origin (or center of) its coordinate system.

As a specific example, consider the following linear Gaussian model:

$$X_n^r = A_n X_{n-1}^r + b_n^r + V_n, \quad n \geq 1 \tag{1}$$

where $V_n$ is zero mean Gaussian additive noise with variance $Q_n$ and $b_n^r$ are deterministic inputs. The measurement $Y_n^r$ made by node $r$ is also defined relative to the local coordinate system at node $r$. For a linear Gaussian observation model the measurement is generated as follows:

$$Y_n^r = C_n^r X_n^r + d_n^r + W_n^r, \quad n \geq 1 \tag{2}$$

where $W_n^r$ is zero mean Gaussian additive noise with variance $R_n^r$ and $d_n^r$ is deterministic. Note that the time varying observation model $\{(C_n^r, d_n^r, R_n^r)\}_{n\geq 1}$ is different for each node. A time-varying state and observation model is retained for an Extended Kalman Filter (EKF) implementation in the non-linear setting to be defined below. It is in this setting that the need for sequences $\{b_n^r\}_{n\geq 1}$ and $\{d_n^r\}_{n\geq 1}$ arises. Also, the dimension of the observation vector $Y_n^r$ need not be the same for different nodes since each node may be equipped with a different sensor type. For example, node $r$ may obtain measurements of the target's position while node $v$ measures bearing. Alternatively, the state-space model in (1)–(2) can be expressed in the form of a Hidden Markov Model (HMM):

$$X_n^r \mid X_{n-1}^r = x_{n-1}^r \sim f_n\left(\cdot \mid x_{n-1}^r\right), \tag{3}$$

$$Y_n^r \mid X_n^r = x_n^r \sim g_n^r\left(\cdot \mid x_n^r\right) \tag{4}$$

where $f_n$ denotes the transition density of the target and $g_n^r$ the density of the likelihood of the observations at node $r$.

Fig. 1(a) illustrates a three node setting where a target is being jointly observed and tracked by three sensors. (Only the position of the target is shown.) At node 1, $X_n^1$ is defined relative to the local coordinate system of node 1 which regards itself as the origin. Similarly for nodes 2 and 3. We define $\theta_*^{i,j}$ to be the position of node $i$ *in the local coordinate system* of node $j$. This means that the vector $X_n^i$ relates to the local coordinate system of node $j$ as follows (see Fig. 1):

$$X_n^j = X_n^i + \theta_*^{i,j}.$$

The *localization* parameters $\{\theta_*^{i,j}\}_{(i,j)\in\mathcal{E}}$ are static as the nodes are not mobile. We note the following obvious but important relationship: if nodes $i$ and $j$ are connected through intermediate nodes $j_1, j_2, \ldots, j_m$ then

$$\theta_*^{i,j} = \theta_*^{i,j_1} + \theta_*^{j_1,j_2} + \theta_*^{j_2,j_3} + \cdots + \theta_*^{j_{m-1},j_m} + \theta_*^{j_m,j}. \tag{5}$$

This relationship is exploited to derive the distributed filtering and localization algorithms in the next section. We define $\theta_*^{i,j}$ so

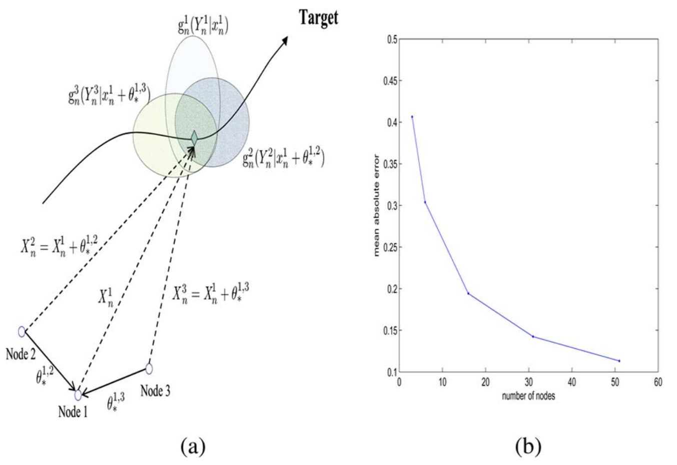

Fig. 1. Left: a three node network tracking a target traversing its field of view. The trajectory of the target is shown with the solid line. Each node regards itself as the center of its local coordinate system. At time $n$ a measurement is registered by all three nodes. The ellipses show the support of the observation densities for the three nodes, i.e., the support of $g_n^1(Y_n^1\,|\cdot)$ is defined as all $x_n^1$ such that $g_n^1(Y_n^1\,|\,x_n^1) > 0$; similarly for the rest. The filtering update step at node 1 will clearly benefit from the observations made by nodes 2 and 3. The localization parameters $\theta_*^{1,2}$, $\theta_*^{1,3}$ are the coordinates of node 1 in the local coordinate systems of node 2 and 3 respectively. While $X_n^r$ was defined to be the state of the target, which includes its velocity, for this illustration only, $X_n^r$ is to be understood as the position of the target at time $n$ w.r.t. the coordinate system of node $r$. Right: Average absolute tracking error is plotted against the number of nodes to illustrate the benefit of collaborative tracking. The results are obtained using a centralized implementation with 50 independent runs, $10^4$ time steps for a chain sensor network of different length and $A_n = B_n = Q_n = C_n^i = D_n^i = R_n^i = 1$, $b_n^i = d_n^i = 0$. (a) Three node joint tracking example. (b) Joint tracking error vs number of nodes.

that the dimensions are the same as the target state vector. When the state vector is comprised of the position and velocity of the target, only the first and third components of $\theta_*^{i,j}$ are relevant while the other two are redundant and set to $\theta_*^{i,j}(2) = 0$ and $\theta_*^{i,j}(4) = 0$. Let

$$\theta_* \equiv \{\theta_*^{i,j}\}_{(i,j)\in\mathcal{E}}, \quad \theta_*^{i,i} \equiv 0 \tag{6}$$

where $\theta_*^{i,i}$ for all $i \in \mathcal{V}$ is defined to be the zero vector.

Let $Y_n$ denote all the measurements received by the network at time $n$, i.e., $Y_n \equiv \{Y_n^v\}_{v\in\mathcal{V}}$. We also denote the sequence $(Y_1,\ldots,Y_n)$ by $Y_{1:n}$. In the *collaborative or joint* filtering problem, each node $r$ computes the local filtering density:

$$p_{\theta_*}^r\left(x_n^r \mid Y_{1:n}\right) \propto p_{\theta_*}^r\left(Y_n \mid x_n^r\right) p_{\theta_*}^r\left(x_n^r \mid Y_{1:n-1}\right) \tag{7}$$

where $p_{\theta_*}^r(x_n^r \mid Y_{1:n-1})$ is the predicted density and is related to the filtering density of the previous time through the following prediction step:

$$\begin{aligned} &p_{\theta_*}^r\left(x_n^r \mid Y_{1:n-1}\right) \\ &\quad = \int f_n\left(x_n^r \mid x_{n-1}^r\right) p_{\theta_*}^r\left(x_{n-1}^r \,\middle|\, Y_{1:n-1}\right) dx_{n-1}^r. \end{aligned} \tag{8}$$

The likelihood term is

$$p_{\theta_*}^r\left(Y_n \mid x_n^r\right) = \prod_{v\in\mathcal{V}} g_n^v\left(Y_n^v \mid x_n^r + \theta_*^{r,v}\right) \tag{9}$$

where the superscript on the densities indicate the coordinate system they are defined w.r.t. (and the node the density belongs to) while the subscript makes explicit the dependence on the localization parameters. Let also $\mu_{n\,|\,n-1}^r$ and $\mu_n^r$ denote the predicted and filtered mean of the densities $p_{\theta_*}^r(x_n^r \mid Y_{1:n-1})$ and $p_{\theta_*}^r(x_n^r \mid Y_{1:n})$, respectively, where the dependence on $\theta_*$ is suppressed in the notation. The prediction step in (8) can be implemented locally at each node without exchange of information, but the update step in (7) incorporates all the measurements of the network. Fig. 1(a) shows the support of the three observation densities as ellipses where the support of $g_n^1(Y_n^1\,|\cdot)$ is defined to be all $x^1$ such that $g_n^1(Y_n^1\,|\cdot) > 0$; similarly for the rest. The filtering update step at node 1 can only include the observations made by nodes 2 and 3 provided the localization parameters $\theta_*^{1,2}$ and $\theta_*^{1,3}$ are known locally to node 1, since the likelihood $p_{\theta_*}^1(Y_n \mid x_n^1)$ defined in (9) is

$$g_n^1\left(Y_n^1 \mid x_n^1\right) g_n^2\left(Y_n^2 \mid x_n^1 + \theta_*^{1,2}\right) g_n^3\left(Y_n^3 \mid x_n^1 + \theta_*^{1,3}\right).$$

The term joint filtering is used since each sensor benefits from the observation made by all the other sensors. An illustration of the benefit w.r.t. the tracking error is in Fig. 1(b). We will show in Section III that it is possible to implement joint filtering in a truly distributed manner, i.e., each node executes a message passing algorithm (with communication limited only to neighboring nodes) that is scalable with the size of the network. However joint filtering hinges on knowledge of the localization parameters $\theta_*$ which are unknown *a priori*. In Section IV we will propose distributed estimation algorithms to learn the localization parameters, which refine the parameter estimates as new data arrive. These proposed algorithms in this context are to the best of our knowledge novel.

## A. Nonlinear Model

Most tracking problems of practical interest are essentially nonlinear non-Gaussian filtering problems. SMC methods, also known as Particle Filters, provide very good approximations to the filtering densities [22]. While it is possible to develop SMC methods for the problem presented here, the resulting algorithms require significantly higher computational cost. We refer the interested reader to ([13, Ch. 9]) for more details. In the interest of execution speed and simplicity, we employ the linearization procedure of the Extended Kalman filter (EKF) when dealing with a non-linear system. Specifically, let the distributed tracking system be given by the following model:

$$X_n^r = \phi_n\left(X_{n-1}^r\right) + V_n, \quad Y_n^r = \psi_n^r\left(X_n^r\right) + W_n^r$$

where $\phi_n : \mathbb{R}^4 \to \mathbb{R}^4$ and $\psi_n^r : \mathbb{R}^4 \to \mathbb{R}^{d_y}$ are smooth continuous functions. At time $n$, each node will linearize its state and observation model about the filtered and predicted mean, respectively. Specifically, a given node $r$ will implement:

$$\begin{aligned} X_n^r &= \phi_n\left(\mu_{n-1}^r\right) \\ &\quad + \nabla\phi_n\left(\mu_{n-1}^r\right)\left(X_{n-1}^r - \mu_{n-1}^r\right) + V_n \end{aligned} \tag{10}$$

$$\begin{aligned} Y_n^r &= \psi_n^r\left(\mu_{n\,|\,n-1}^r\right) \\ &\quad + \nabla\psi_n^r\left(\mu_{n\,|\,n-1}^r\right)\left(X_n^r - \mu_{n\,|\,n-1}^r\right) + W_n^r. \end{aligned} \tag{11}$$

where for a mapping $f : \mathbb{R}^d \to \mathbb{R}^d$, $\nabla f \equiv [\nabla f_1, \ldots, \nabla f_d]^{\mathrm{T}}$. Note that after linearization extra additive terms appear as seen in the setting described by (1)–(2).

### *B. Message Passing*

Assume at time $n$, the estimate of the localization parameters is $\theta_n = \{\theta_n^{i,j}\}_{(i,j)\in\mathcal{E}}$, with $\theta_n^{i,j}$ known to node $j$ only. To perform the prediction and update steps in (7)–(8) locally at each node a naive approach might require each node to access to all localization parameters $\theta_n$ and all the different model parameters $\{(C_n^r, d_n^r, R_n^r)\}_{n\geq 1, r\in\mathcal{V}}$. A scheme that requires all this information to be passed at every node would be inefficient due to excessive communication and redundant computations. The core idea in this paper is to avoid this by storing the parameters in $\theta_n$ across the network and perform required computations only at the nodes where the parameters are stored. The results of these computations are then propagated in the network using an efficient message passing scheme.

**Algorithm 1:** Generic message passing at time $n$

```
At $k = 1$, compute:
    $m_{n,1}^{i,j} = F_n^i,$    (12)
    $\ddot{m}_{n,1}^{i,j} = F_n^i \theta_n^{j,i}.$    (13)
for $k = 2, \ldots, K$ compute:
    $m_{n,k}^{i,j} = F_n^i + \sum_{p\in \mathrm{ne}(i)\setminus\{j\}} m_{n,k-1}^{p,i},$    (14)
    $\ddot{m}_{n,k}^{i,j} = m_{n,k}^{i,j}\theta_n^{j,i} + \sum_{p\in \mathrm{ne}(i)\setminus\{j\}} \ddot{m}_{n,k-1}^{p,i}.$    (15)
end for
```

Message passing is an iterative procedure with $k = 1, \ldots, K$ iterations for each time $n$ and is steered towards the development of a distributed Kalman filter, whose presentation is postponed for the next section. In Algorithm 1 we define a recursion of messages which are to be communicated between all pairs of neighboring nodes in both directions. Here $\mathrm{ne}(i)$ denote the neighbors of node $i$ excluding node $i$ itself. At iteration $k$ the computed messages from node $i$ to $j$ are matrix and vector quantities of appropriate dimensions and are denoted as $m_{n,k}^{i,j}$ and $\ddot{m}_{n,k}^{i,j}$ respectively. The source node is indicated by the first letter of the superscript. Note that during the execution of Algorithm 1 time $n$ remains fixed and iteration $k$ should not be confused with time $n$. Clearly we assume that the sensors have the ability to communicate much faster than collecting measurements. We proceed with a simple (but key) lemma concerning the aggregations of sufficient statistics locally at each node.

*Lemma 1:* At time $n$, let $\{F_n^v\}_{v\in\mathcal{V}}$ be a collection of matrices where $F_n^v$ is known to node $v$ only, and consider the task of computing $\sum_{v\in\mathcal{V}} F_n^v$ and $\sum_{v\in\mathcal{V}} F_n^v \theta_n^{r,v}$ at each node $r$ of a network with a tree topology. Using Algorithm 1 and if $K$ is at least as large as the number of edges connecting the two farthest nodes in the network, then $\sum_{v\in\mathcal{V}} F_n^v = F_n^r + \sum_{j\in\mathrm{ne}(r)} m_{n,K}^{j,r}$ and $\sum_{v\in\mathcal{V}} F_n^v\theta_n^{r,v} = \sum_{j\in\mathrm{ne}(r)} \ddot{m}_{n,K}^{j,r}$.

(The proof, which uses (5), is omitted.) An additional advantage here is that if the network is very large, in the interest of speed one might be interested in settling with computing the presented sums only for a subset of nodes and thus use a smaller $K$. This also applies when a target traverses the field of view of the sensors swiftly and is visible only by few nodes at each time. Finally, a lower value for $K$ is also useful when cycles are present in order to avoid summing each $F_n^i$ more than once, albeit summing only over a subset of $\mathcal{V}$.

## III. Distributed Joint Filtering

For a linear Gaussian system, the joint filter $p_\theta^r(x_n^r \mid Y_{1:n})$ at node $r$ is a Gaussian distribution with a specific mean vector $\mu_n^r$ and covariance matrix $\Sigma_n^r$. The derivation of the Kalman filter to implement $p_\theta^r(x_n^r \mid Y_{1:n})$ is standard upon noting that the measurement model at node $r$ can be written as $Y_n = C_n X_n^r + d_n + W_n$ where the $i$th block of $Y_n$, $Y_n^i$, satisfies $Y_n^i = C_n^i(X_n^r + \theta^{r,i}) + d_n^i + W_n^i$. However, there will be "non-local" steps due to the requirement that quantities $\sum_{i\in\mathcal{V}}(C_n^i)^{\mathrm{T}}(R_n^i)^{-1}C_n^i$, $\sum_{i\in\mathcal{V}}(C_n^i)^{\mathrm{T}}(R_n^i)^{-1}Y_n^i$ and $\sum_{i\in\mathcal{V}}(C_n^i)^{\mathrm{T}}(R_n^i)^{-1}C_n^i\theta^{r,i}$ be available locally at node $r$. To solve this problem, we may use Lemma 1 with $F_n^i = (C_n^i)^{\mathrm{T}}(R_n^i)^{-1}C_n^i$ and in order to compute $\sum_{i\in\mathcal{V}}(C_n^i)^{\mathrm{T}}(R_n^i)^{-1}Y_n^i$ we will define $\dot{m}_{n,k}^{i,j}$ that is an additional message similar to $m_{n,k}^{i,j}$.

Recall that $b_n^i, d_n^i$ are known local variables that arose due to linearization. Also to aid the development of the distributed on-line localization algorithms in Section IV, we assume that for the time being the localization parameter estimates $\{\theta_n\}_{n\geq 1}$ are time-varying and known to the relevant nodes they belong. For the case where that $b_n^i, d_n^i = 0$, we summarize the resulting distributed Kalman filter in Algorithm 2, which is to be implemented at every node of the network. Note that messages (16)–(18) are matrix and vector valued quantities and require a fixed amount of memory regardless of the number of nodes in the network. Also, the same rule for generating and combining messages are implemented at each node. The distributed Kalman filter presented here bears a similar structure to the one found in [23].

In the case $b_n^i, d_n^i \neq 0$ modifications to Algorithm 2 are as follows: in (19), to the right hand side of $\mu_{n\mid n-1}^r$, the term $b_n^r$ should be added and all instances of $Y_n^r$ should be replaced with $Y_n^r - d_n^r$. Therefore the assuming $b_n^i, d_n^i = 0$ does not compromise the generality of the approach. A direct application of this modification is the distributed EKF, which is obtained by adding the term $\phi_n(\mu_{n-1}^r) - \nabla\phi_n(\mu_{n-1}^r)\mu_{n-1}^r$ to the right hand side of $\mu_{n\mid n-1}^r$ in (19), and replacing all instances of $Y_n^r$ with $Y_n^r - \psi_n^r(\mu_{n\mid n-1}^r) + \nabla\psi_n^r(\mu_{n\mid n-1}^r)\mu_{n\mid n-1}^r$. In addition, one needs to replace $A_n$ with $\nabla\phi_n(\mu_{n-1}^r)$.

## IV. Distributed Collaborative Localization

Following the discussion in Section II we will treat the sensor localization problem as a static parameter estimation problem for HMMs. The purpose of this section is to develop a fully decentralized implementation of popular Maximum Likelihood (ML) techniques for parameter estimation in HMMs. We will focus on two on-line ML estimation methods: Recursive Maximum Likelihood (RML) and Expectation-Maximization (EM).

**Algorithm 2:** Distributed Filtering

```
1: for $n \geq 1$:
2:     Let the localization parameter be $\theta_n$ and the set of collected measurements be $Y_n = \{Y_n^v\}_{v \in \mathcal{V}}$. Initialize messages $(m_{n,k}^{i,j}, \dot{m}_{n,k}^{i,j}, \ddot{m}_{n,k}^{i,j})$ and $(m_{n,k}^{j,i}, \dot{m}_{n,k}^{j,i}, \ddot{m}_{n,k}^{j,i})$ for all neighboring nodes $(i,j) \in \mathcal{E}$ as:
           $m_{n,1}^{i,j} = (C_n^i)^{\mathrm{T}} (R_n^i)^{-1} C_n^i,$
           $\dot{m}_{n,1}^{i,j} = (C_n^i)^{\mathrm{T}} (R_n^i)^{-1} Y_n^i,$
           $\ddot{m}_{n,1}^{i,j} = m_n^{i,j} \theta_n^{j,i}.$
3:     for $k = 2, \ldots, K$ exchange the messages $(m_{n,k}^{i,j}, \dot{m}_{n,k}^{i,j}, \ddot{m}_{n,k}^{i,j})$ and $(m_{n,k}^{j,i}, \dot{m}_{n,k}^{j,i}, \ddot{m}_{n,k}^{j,i})$ defined below between all neighboring nodes $(i,j) \in \mathcal{E}$:
           $m_{n,k}^{i,j} = (C_n^i)^{\mathrm{T}} (R_n^i)^{-1} C_n^i + \sum_{p \in \mathrm{ne}(i) \backslash \{j\}} m_{n,k-1}^{p,i},$   (16)
           $\dot{m}_{n,k}^{i,j} = (C_n^i)^{\mathrm{T}} (R_n^i)^{-1} Y_n^i + \sum_{p \in \mathrm{ne}(i) \backslash \{j\}} \dot{m}_{n,k-1}^{p,i},$   (17)
           $\ddot{m}_{n,k}^{i,j} = m_n^{i,j} \theta_n^{j,i} + \sum_{p \in \mathrm{ne}(i) \backslash \{j\}} \ddot{m}_{n,k-1}^{p,i}.$   (18)
4:     end for
5:     Update the local filtering densities at each node $r \in \mathcal{V}$:
           $\mu_{n\,|\,n-1}^r = A_n \mu_{n-1}^r, \Sigma_{n\,|\,n-1}^r = A_n \Sigma_{n-1}^r A_n^{\mathrm{T}} + Q_n,$   (19)
           $M_n^r = \left(\Sigma_{n\,|\,n-1}^r\right)^{-1} + (C_n^r)^{\mathrm{T}} (R_n^r)^{-1} C_n^r + \sum_{i \in \mathrm{ne}(r)} m_n^{i,r},$   (20)
           $z_n^r = \left(\Sigma_{n\,|\,n-1}^r\right)^{-1} \mu_{n\,|\,n-1}^r + (C_n^r)^{\mathrm{T}} (R_n^r)^{-1} Y_n^r + \sum_{i \in \mathrm{ne}(r)} \left(\dot{m}_n^{i,r} - \ddot{m}_n^{i,r}\right),$   (21)
           $\Sigma_n^r = (M_n^r)^{-1}, \quad \mu_n^r = \Sigma_n^r z_n^r.$   (22)
6: end for
```

The core idea in our distributed ML formulation is to store the parameter $\theta_n = \{\theta_n^{i,j}\}_{(i,j) \in \mathcal{E}}$ across the network. Each node $r$ will use the available data $Y_{1:n}$ from every node to estimate $\theta_*^{r,j}$, which is the component of $\theta_*$ corresponding to edge $(r,j)$. This can be achieved computing at each node $r$ the ML estimate:

$$\tilde{\theta}_n^{r,j} = \arg \max_{\theta^{r,j} \in \mathbb{R}^4} \log p_\theta^r (Y_{1:n}). \tag{23}$$

Note that each node maximizes its "local" likelihood function although all the data across the network is being used.

On-line parameter estimation techniques like the RML and on-line EM are suitable for sensor localization in surveillance applications because we expect a practically indefinite length of observations to arrive sequentially. For example, objects will persistently traverse the field of view of these sensors, i.e., the departure of old objects would be replenished by the arrival of new ones. A recursive procedure is essential to give a quick up-to-date parameter estimate every time a new set of observations is collected by the network. This is done by allowing every node $r$ to update the estimate of the parameter along edge $(r,j)$, $\theta_n^{r,j}$, according to a rule like

$$\theta_{n+1}^{r,j} = G_{n+1}^{r,j}(\theta_n, Y_n), \quad n \geq 1 \tag{24}$$

where $G_{n+1}^{r,j}$ is an appropriate function to be defined. Similarly each neighbor $j$ of $r$ will perform a similar update along the same edge only this time it will update $\theta_n^{j,r}$.

## *A. Distributed RML*

For distributed RML, each node $r$ updates the parameter of edge $(r,j)$ using

$$\theta_{n+1}^{r,j} = \theta_n^{r,j} + \gamma_{n+1}^r \nabla_{\theta^{r,j}} \log \int p_\theta^r (Y_n \mid x_n^r) \, p_\theta^r \times (x_n^r \mid Y_{1:n-1}) \, dx_n^r \Big|_{\theta = \theta_n} \tag{25}$$

where $\gamma_{n+1}^r$ is a step-size that should satisfy $\sum_n \gamma_n^r = \infty$ and $\sum_n (\gamma_n^r)^2 < \infty$.

The gradient in (25) is w.r.t. $\theta^{r,j}$. The local joint *predicted* density $p_\theta^r(x_n^r \mid Y_{1:n-1})$ at node $r$ was defined in (8) and is a function of $\theta = \{\theta^{i,j}\}_{(i,j) \in \mathcal{E}}$, and likelihood term is given in (9). Also, the gradient is evaluated at $\theta_n = \{\theta_n^{i,j}\}_{(i,j) \in \mathcal{E}}$ while only $\theta_n^{r,j}$ is available locally at node $r$. The remaining values $\theta_n$ are stored across the network. All nodes of the network will implement such a local gradient algorithm with respect to the parameter associated to its adjacent edge. We note that (25) in the present form is not an on-line parameter update like (24) as it requires browsing through the entire history of observations. This limitation is removed by defining certain intermediate quantities that facilitate the online evaluation of this gradient in the spirit of [17], [18] (see in the Appendix for more details).

**Algorithm 3:** Distributed RML

**for** $n \geq 1$: let the current parameter estimate be $\theta_n$. Upon obtaining measurements $Y_n = \{Y_n^v\}_{v \in \mathcal{V}}$ the following filtering and parameter update steps are to be performed.

**Filtering step**: Perform steps (2–5) in Algorithm 2.

**Parameter update**: Each node $r \in \mathcal{V}$ of the network will update the following quantities for every edge $(r,j) \in \mathcal{E}$:

$$\dot{\mu}_{n\,|\,n-1}^{r,j} = A_n \dot{\mu}_{n-1}^{r,j}, \tag{26}$$

$$\dot{z}_n^{r,j} = \left(\Sigma_{n\,|\,n-1}^r\right)^{-1} \dot{\mu}_{n\,|\,n-1}^{r,j} - m_{n,K}^{j,r}, \tag{27}$$

$$\dot{\mu}_n^{r,j} = (M_n^r)^{-1} \dot{z}_n^{r,j}. \tag{28}$$

Upon doing so the localization parameter is updated:

$$\theta_{n+1}^{r,j} = \theta_n^{r,j} + \gamma_{n+1}^r \left[ -\left(\dot{\mu}_{n\,|\,n-1}^{r,j}\right)^{\mathrm{T}} \left(\Sigma_{n\,|\,n-1}^r\right)^{-1} \mu_{n\,|\,n-1}^r + \left(\dot{z}_n^{r,j}\right)^{\mathrm{T}} (M_n^r)^{-1} z_n^r + \dot{m}_{n,K}^{j,r} - \ddot{m}_{n,K}^{j,r} \right].$$

**end for**

The distributed RML implementation for self-localization and tracking is presented in Algorithm 3, while the derivation of the algorithm is presented in the Appendix. The intermediate quantities (26)–(28) take values in $\mathbb{R}^{4\times 2}$ and may be initialized to zero matrices. For the non-linear model, when an EKF implementation is used for Algorithm 2, then Algorithm 3 remains the same.

## B. Distributed On-Line EM

We begin with a brief description of distributed EM in an off-line context and then present its on-line implementation. Given a batch of $T$ observations, let $p$ be the (off-line) iteration index and $\theta_p = \{\theta_p^{i,j}\}_{(i,j)\in\mathcal{E}}$ be the current estimate of $\theta_*$ after $p-1$ distributed EM iterations on the batch of observations $Y_{1:T}$. Each edge controlling node $r$ will execute the following E and M steps to update the estimate of the localization parameter for its edge. For iteration $p = 1, 2, \ldots$

$$\text{(E) } Q^r(\theta_p,\theta) = \int \log p_\theta^r\left(x_{1:T}^r, Y_{1:T}\right) p_{\theta_p}^r\left(x_{1:T}^r \mid Y_{1:T}\right) dx_{1:T}^r,$$

$$\text{(M) } \theta_{p+1}^{r,j} = \arg\max_{\theta^{r,j}} \quad Q^r\left(\theta_p, \left(\theta^{r,j}, \theta_p^{-(r,j)}\right)\right),$$

where $\theta_p^{-(r,j)} = \{\theta_p^e\}_{e\in\mathcal{E}\setminus(r,j)}$.

To show how the E-step can be computed we write $p_\theta^r(x_{1:T}^r, Y_{1:T})$ as

$$p_\theta^r\left(x_{1:T}^r\right) p_\theta^r\left(Y_{1:T} \mid x_{1:T}^r\right) = \prod_{n=1}^T f_n\left(x_n^r \mid x_{n-1}^r\right) p_\theta^r\left(Y_n \mid x_n^r\right)$$

where $p_\theta^r(Y_n \mid x_n^r)$ was defined in (9). Note that $p_{\theta_p}^r(x_{1:T}^r \mid Y_{1:T})$ is a function of $\theta_p = \{\theta_p^{i,i'}\}_{(i,i')\in\mathcal{E}}$ (and not just $\theta_p^{r,j}$) and the $\theta$-dependence of $p_\theta^r(x_{1:T}^r, Y_{1:T})$ arises through the likelihood term only as $p_\theta^r(x_{1:T}^r)$ is $\theta$-independent. This means that in order to compute the E-step, it is sufficient to maintain the smoothed marginals: $p_\theta^r(x_n^r \mid Y_{1:T}) \propto \int p_\theta^r(x_{1:T}^r, Y_{1:T})\, dx_{1:T\setminus\{n\}}^r$, where $1 \le n \le T$ and $dx_{1:T\setminus\{n\}}^r$ means integration w.r.t. all variables except $x_n^r$. For linear Gaussian models this smoothed density is also Gaussian, with its mean and covariance denoted by $\mu_{n\mid T}^r, \Sigma_{n\mid T}^r$ respectively.

The M-step is solved by setting the derivative of $Q^r(\theta_p, (\theta^{r,j}, \theta_p^{-(r,j)}))$ w.r.t. $\theta^{r,j}$ to zero. The details are presented in the Appendix and the main result is:

$$\nabla_{\theta^{r,j}} \int \log p_\theta^r\left(Y_n \mid x_n^r\right) p_{\theta_p}^r\left(x_n^r \mid Y_{1:T}\right) dx_n^r = \dot{m}_{n,K}^{j,r} - \ddot{m}_{n,K}^{j,r} - \left(m_{n,K}^{j,r}\right)^{\mathrm{T}} \mu_{n\mid T}^r,$$

where $(m_{n,K}^{j,r}, \dot{m}_{n,K}^{j,r}, \ddot{m}_{n,K}^{j,r})$, defined in (16)–(18), are propagated with localization parameter $\theta_p$ for all observations from time 1 to $T$. Only $\ddot{m}_{n,K}^{j,r}$ is a function of $\theta^{r,j}$. To perform the M-step, the following equation is solved for $\theta^{r,j}$

$$\left(\sum_{n=1}^T m_{n,K}^{j,r}\right)\theta^{r,j} = \sum_{n=1}^T \left(\dot{m}_{n,K}^{j,r} - \left(m_{n,K}^{j,r}\right)^{\mathrm{T}} \mu_{n\mid T}^r - \dot{m}_{n,K}^{j,r} + \ddot{m}_{n,1}^{j,r}\right). \quad (29)$$

Note that $\theta^{r,j}$ is a function of quantities available locally to node $r$ only. The M-step can also be written as the following function:

$$\Lambda\left(\mathcal{S}_{T,1}^{r,j}, \mathcal{S}_{T,2}^{r,j}, \mathcal{S}_{T,3}^{r,j}\right) = \left(\mathcal{S}_{T,2}^{r,j}\right)^{-1}\left(\mathcal{S}_{T,3}^{r,j} - \mathcal{S}_{T,1}^{r,j}\right)$$

where $\mathcal{S}_{T,1}^{r,j}, \mathcal{S}_{T,2}^{r,j}, \mathcal{S}_{T,3}^{r,j}$ are three summary statistics of the form:

$$\mathcal{S}_{T,l}^{r,j} = \frac{1}{T}\int \left(\sum_{n=1}^T s_{n,l}^{r,j}\left(x_n^r, Y_n\right)\right) p_{\theta_p}^r\left(x_n^r \mid Y_{1:T}\right) dx_n^r, \qquad l = 1, 2, 3,$$

with $s_{n,l}^{r,j}$ being defined as follows:

$$s_{n,1}^{r,j}\left(x_n^r, Y_n\right) = \left(m_{n,K}^{j,r}\right)^{\mathrm{T}} x_n^r, \quad s_{n,2}^{r,j}\left(x_n^r, Y_n\right) = m_{n,K}^{j,r} \quad (30)$$

$$s_{n,3}^{r,j}\left(x_n^r, Y_n\right) = \dot{m}_{n,K}^{j,r} - \ddot{m}_{n,K}^{j,r} + \ddot{m}_{n,1}^{j,r}. \quad (31)$$

Note that for this problem $s_{n,2}^{r,j}$ and $s_{n,3}^{r,j}$ are state independent.

An on-line implementation of EM follows by computing recursively running averages for each of the three summary statistics, which we will denote as $\mathcal{S}_{n,1}^{r,j}, \mathcal{S}_{n,2}^{r,j}, \mathcal{S}_{n,3}^{r,j}$. At each time $n$ these will be used at every node $r$ to update $\theta^{r,j}$ using $\theta_{n+1}^{r,j} = \Lambda(\mathcal{S}_{n,1}^{r,j}, \mathcal{S}_{n,2}^{r,j}, \mathcal{S}_{n,3}^{r,j})$. Note that $\Lambda$ is the same function for every node. The on-line implementation of distributed EM is found in Algorithm 4. All the steps are performed with quantities available locally at node $r$ using the exchange of messages as detailed in Algorithm 2. Here $\gamma_n^r$ is a step-size satisfying the same conditions as in RML and $\theta_0$ can be initialized arbitrarily, e.g., the zero vector. Finally, it has been reported in [24] that it is usually beneficial for the first few epochs not to perform the M step in (32) and allow a burn-in period for the running averages of the summary statistics to converge.

**Algorithm 4:** Distributed on-line EM

**for** $n \ge 1$: let the current parameter estimate be $\theta_n$. Upon obtaining measurements $Y_n = \{Y_n^v\}_{v\in\mathcal{V}}$ the following filtering and parameter update steps are to be performed.

**Filtering step**: Perform steps (2–5) in Algorithm 2. Also compute $\tilde{\Sigma}_n^r = (\Sigma_{n-1}^r + A_n^{\mathrm{T}} Q_n^{-1} A_n)^{-1}$.

**Parameter update**: Each node $r \in \mathcal{V}$ of the network will update the following quantities for every edge $(r,j) \in \mathcal{E}$:

$$H_n^{r,j} = \gamma_n^r\left(m_{n,K}^{j,r}\right)^{\mathrm{T}} + (1-\gamma_n^r) H_{n-1}^{r,j}\left(\tilde{\Sigma}_n^r\right)^{-1} A_n^{\mathrm{T}} Q_n^{-1},$$

$$h_n^{r,j} = (1-\gamma_n^r)\left(H_{n-1}^{r,j}\left(\tilde{\Sigma}_n^r\right)^{-1}\left(\Sigma_{n-1}^r\right)^{-1}\mu_{n-1}^r + h_{n-1}^{r,j}\right),$$

$$\mathcal{S}_{n,1}^{r,j} = H_n^{r,j}\mu_n^r + h_n^{r,j},$$

$$\mathcal{S}_{n,2}^{r,j} = \gamma_n^r m_{n,K}^{j,r} + (1-\gamma_n^r)\mathcal{S}_{n-1,2}^{r,j},$$

$$\mathcal{S}_{n,3}^{r,j} = \gamma_n^r\left(\dot{m}_{n,K}^{j,r} - \ddot{m}_{n,K}^{j,r} + \ddot{m}_{n,1}^{j,r}\right) + (1-\gamma_n^r)\mathcal{S}_{n-1,3}^{r,j}.$$

Upon doing so the localization parameter is updated:

$$\theta_{n+1}^{r,j} = \Lambda\left(\mathcal{S}_{n,1}^{r,j}, \mathcal{S}_{n,2}^{r,j}, \mathcal{S}_{n,3}^{r,j}\right). \quad (32)$$

**end for**

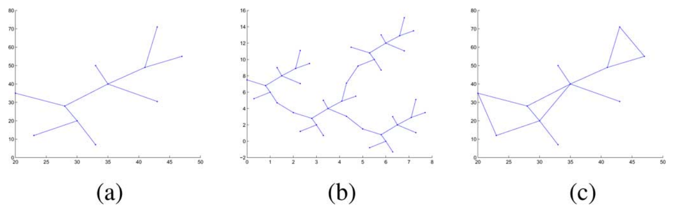

Fig. 2. Various sensor networks of different size and topology. (a) 11 node sensor network, (b) 44 node sensor network, (c) 11 node sensor network with cycles.

## V. Numerical Examples

The performance of the distributed RML and EM algorithms are studied using a Linear Gaussian and a non-linear model. For both cases the hidden target is given in (1) with $V_n = B\tilde{V}_n$, where $\tilde{V}_n$ is zero mean Gaussian additive noise with variance $\tilde{Q}_n$, and

$$A_n = \begin{bmatrix} 1 & \tau & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{\tau^2}{2} & 0 \\ \tau & 0 \\ 0 & \frac{\tau^2}{2} \\ 0 & \tau \end{bmatrix}, \quad \tilde{Q}_n = \sigma_x^2 I$$

and $I$ is the identity matrix. For the linear model the observations are given by (2) with

$$C_n^r = \alpha^r \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad R_n^r = \sigma_y^2 I,$$

where $\alpha^r$ are constants different for each node and are assigned randomly from the interval [0.75, 1.25]. For the non-linear model we will use the bearings-only measurement model. In this model at each node $r$, the observation $Y_n^r$ is:

$$Y_n^r = \tan^{-1}\left(X_n^r(1)/X_n^r(3)\right) + W_n^r.$$

with $W_n^r \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 0.35^2)$. For the remaining parameters we set $\tau = 0.01$, $\sigma_x = 1$ and $\theta_0^{r,j} = 0$ for all $(r,j) \in \mathcal{E}$. In Fig. 2 we show three different sensor networks for which we will perform numerical experiments.

In Fig. 3 we present various convergence plots for each of these networks for $\sigma_y = 0.5$. We plot both dimensions of the errors $\theta_*^{r,j} - \theta_n^{r,j}$ for three cases:

- in (a) and (d) we use distributed RML and on-line EM respectively for the network of Fig. 2(a) and the linear Gaussian model.
- in (b) and (e) we use distributed RML for the bearings only tracking model and the networks of Fig. 2(a) and (b), respectively. Local linearization as discussed in Sections II-A, III, and IV-A was used to implement the distributed RML algorithm. We remark that we do not apply the online EM to problems where the solution to the M-step cannot be expressed analytically as some function $\Lambda$ of summary statistics.
- in (c) and (f) we use distributed RML and on-line EM for respectively for the network of Fig. 2(c) and the linear Gaussian model. In this case we used $K = 2$.

All errors converge to zero. Although both methods are theoretically locally optimal when performing the simulations we did not observe significant discrepancies in the errors for different initializations. For both RML and on-line EM we used for $n \leq 10^3$ a constant but small step-size, $\gamma_n^r = \gamma = 4 \times 10^{-3}$ and 0.025 respectively. For the subsequent iterations we set $\gamma_n^r = \gamma(n - 10^3)^{-0.8}$. Note that if the step-size decreases too quickly in the first time steps, these algorithms might converge too slowly. In the plots of Fig. 3 one can notice that the distributed RML and EM algorithms require comparable amount of time to converge with the RML being usually faster, with the converge rate depending on the network, the value of $K$ and the simulation parameters.

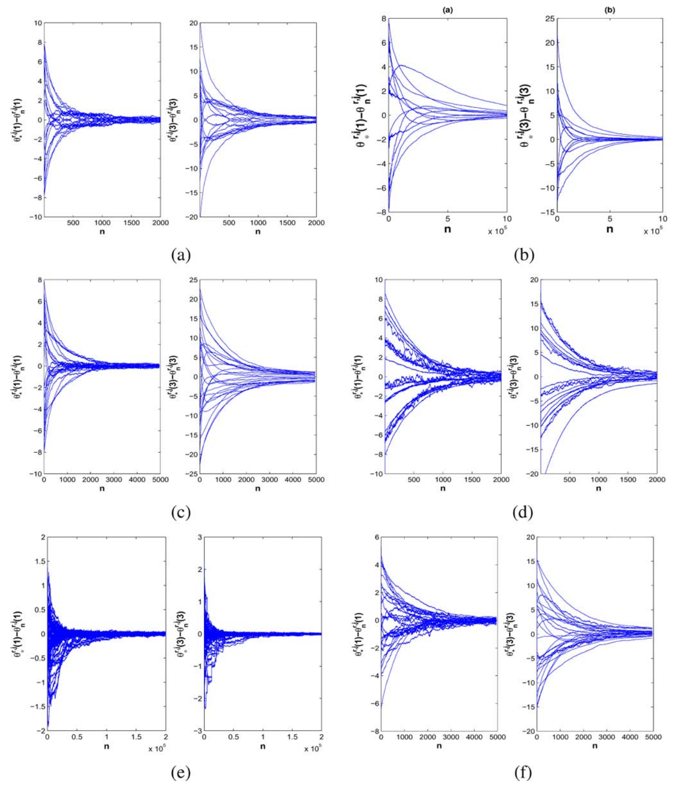

Fig. 3. The convergence of the localization parameters' estimates to $\theta_*^{r,j}$ is demonstrated using appropriate error plots for various sensor networks. Left: Parameter error after each iteration for each edge of the medium sensor network of Fig. 2(a). In each subfigure left and right columns show the errors in the x- and y- coordinates respectively; (a) is for RML and (d) is for EM. Middle: Same errors when using RML for the nonlinear bearings-only observation model; (b) is for medium sized network of Fig. 2(a) and (e) for the large network of Fig. 2(b). Right: Same errors for network with cycles seen in Fig. 2(c); (c) for RML and (f) for EM. (a) RML for tree network. (b) Nonlinear RML for tree network. (c) RML for network with cycles. (d) EM for tree network. (e) Nonlinear RML for large network. (f) EM for network with cycles.

To investigate this further we varied $K$ and $\frac{\sigma_x}{\sigma_y}$ and recorded the root mean squared error (RMSE) for $\theta_n$ obtained for the network of Fig. 2(b) using 50 independent runs. For the RMSE at time $n$ we will use $\sqrt{\frac{1}{50|\mathcal{E}|}\sum_{e\in\mathcal{E}}\sum_{m=1}^{50}\|\theta_*^{r,j} - \theta_{n,m}^{r,j}\|_2^2}$, where $\theta_{n,m}^{r,j}$ denotes the estimated parameter at epoch $n$ obtained from the $m$th run. The results are plotted in Fig. 4 for different cases:

- in (a) and (b) for $\frac{\sigma_x}{\sigma_y} = 2$ we show the RMSE for $K = 2, 4, 8, 12$. We observe that in every case the RMSE keeps reducing as $n$ increases. Both algorithms behave similarly with the RML performing better and showing quicker convergence. One expects that observations beyond your near

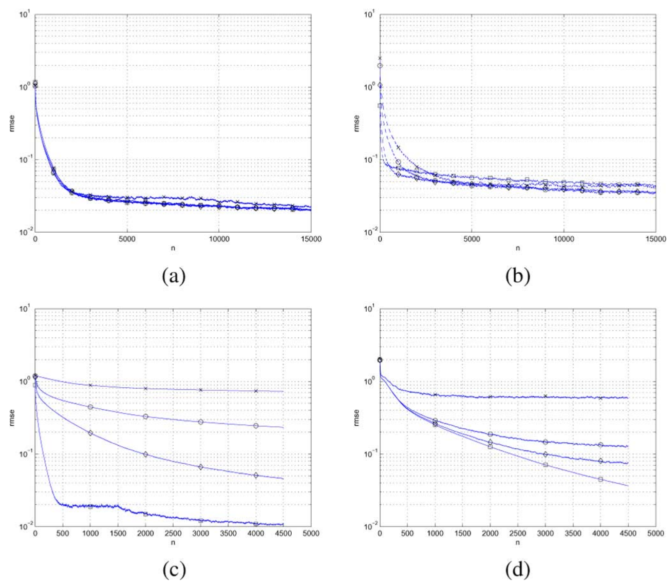

Fig. 4. Comparison of distributed RML and on-line EM. (a) (and (b) resp.): RMSE for RML (and on-line EM resp.) against $n$ for $K = 2\ (\square), 4\ (\diamond), 8\ (\circ), 12\ (\times)$. (c) (and (d) resp.): RMSE for RML (and on-line EM resp.) for $\frac{\sigma_x}{\sigma_y} = 10\ (\square), 1\ (\diamond), 0.5\ (\circ), 0.1\ (\times)$. (a) RML for different K. (b) EM for different K. (c) RML for different $\frac{\sigma_x}{\sigma_y}$. (d) EM for different $\frac{\sigma_x}{\sigma_y}$.

immediate neighbors are not necessary to localize adjacent nodes and hence the good performance for small values of $K$.

- in (b) and (c) we show the RMSE for RML and on-line EM respectively when $\frac{\sigma_x}{\sigma_y} = 10, 1, 0.5, 0.1$. We observe that EM seems to be slightly more accurate for lower values of $\frac{\sigma_x}{\sigma_y}$ with the reverse holding for higher values of the ratio.

In each run the same step-size was used as before except for RML and $\frac{\sigma_x}{\sigma_y} = 10$, where we had to reduce the step size by a factor of 10.

## VI. Conclusion

We have presented two distributed ML methods to perform collaborative tracking and self-localization. We exploited the fact that different nodes collect measurements of a common target. The methods rely on a message passing scheme, which provides the appropriate summary statistics for computing the filtering density and performing parameter updates. There is good empirical evidence that the distributed implementations of ML proposed in this paper are stable and do seem to settle at reasonably accurate estimates. A theoretical investigation of the properties of the schemes would be an interesting but challenging extension. Finally, as pointed out by one referee, another interesting extension would be to develop consensus versions of Algorithm 1 in the spirit of gossip algorithms in [25] or the aggregation algorithm of [26] which might be particularly relevant for networks with cycles, which are dealt with here by using an appropriate value for $K$.

## Appendix A
## Distributed RML Derivation

Let $\theta_n = \{\theta_n^{i,j}\}_{(i,j)\in\mathcal{E}}$ be the estimate of the true parameter $\theta_*$ given the available data $Y_{1:n-1}$. Consider an arbitrary node $r$ and assume it controls edge $(r,j)$. At time $n$, we assume the following quantities are available: $(\dot{\mu}_{n-1}^{r,j} = \nabla_{\theta^{r,j}} \mu_{n-1}^r|_{\theta=\theta_n}, \mu_{n-1}^r|_{\theta=\theta_n}, \Sigma_{n-1}^r)$. The first of these quantities is the derivative of the conditional mean of the hidden state at node $r$ given $Y_{1:n-1}$, i.e., $\nabla_{\theta^{r,j}} \int x_{n-1}^r p_\theta^r(x_{n-1}^r \,|\, Y_{1:n-1})\, dx_{n-1}^r|_{\theta=\theta_n}$. This quantity is a function of the localization parameter $\theta_n$. $\Sigma_{n-1}^r$ is the variance of the distribution $p_\theta^r(x_{n-1}^r \,|\, Y_{1:n-1})|_{\theta=\theta_n}$ and is independent of the localization parameter. The log-likelihood in (25) evaluates to

$$\begin{aligned}\log p_\theta^r(Y_n \,|\, Y_{1:n-1}) &= -\frac{1}{2}\sum_{i\in\mathcal{V}} \left(Y_n^i - C_n^i\theta^{r,i}\right)^T R_n^{i-1}\left(Y_n^i - C_n^i\theta^{r,i}\right)\\ &\quad -\frac{1}{2}\mu_{n\,|\,n-1}^{r\ \mathrm{T}}\left(\Sigma_{n\,|\,n-1}^r\right)^{-1}\mu_{n\,|\,n-1}^r\\ &\quad +\frac{1}{2}\left(z_n^r\right)^{\mathrm{T}}\left(M_n^r\right)^{-1} z_n^r + \text{const}\end{aligned}$$

where all $\theta$ independent terms have been lumped together in the term 'const'. (Refer to Algorithm 2 for the definition of the quantities in this expression.) Differentiating this expression w.r.t. $\theta^{r,j}$ yields

$$\begin{aligned}\nabla_{\theta^{r,j}} \log p_\theta^r(Y_n \,|\, Y_{1:n-1}) &= -\left(\nabla_{\theta^{r,j}}\mu_{n\,|\,n-1}^r\right)^{\mathrm{T}}\left(\Sigma_{n\,|\,n-1}^r\right)^{-1}\mu_{n\,|\,n-1}^r\\ &\quad + \left(\nabla_{\theta^{r,j}} z_n^r\right)^{\mathrm{T}}\left(M_n^r\right)^{-1} z_n^r\\ &\quad + \sum_{i\in\mathcal{V}}\left(\nabla_{\theta^{r,j}}\theta^{r,i}\right)^{\mathrm{T}}\left(C_n^i\right)^{\mathrm{T}}\left(R_n^i\right)^{-1}\left(Y_n^i - C_n^i\theta^{r,i}\right).\end{aligned}$$

(25) requires $\nabla_{\theta^{r,j}} \log p_\theta^r(Y_n \,|\, Y_{1:n-1})$ to be evaluated at $\theta = \theta_n$. Using the (19)–(22) and the assumed knowledge of $(\dot{\mu}_{n-1}^{r,j}, \mu_{n-1}^r|_{\theta=\theta_n}, \Sigma_{n-1}^r)$ we can evaluate the derivatives on the right-hand side of this expression:

$$\dot{\mu}_{n\,|\,n-1}^{r,j} = \nabla_{\theta^{r,j}}\mu_{n\,|\,n-1}^r\Big|_{\theta=\theta_n} = A_n\dot{\mu}_{n-1}^{r,j}, \tag{33}$$

$$\begin{aligned}\dot{z}_n^{r,j} &= \nabla_{\theta^{r,j}} z_n^r|_{\theta=\theta_n} = \left(\Sigma_{n\,|\,n-1}^r\right)^{-1}\dot{\mu}_{n\,|\,n-1}^{r,j}\\ &\quad - \sum_{i\in\mathcal{V}}\left(C_n^i\right)^{\mathrm{T}}\left(R_n^i\right)^{-1} C_n^i\, \nabla_{\theta^{r,j}}\theta^{r,i}\big|_{\theta=\theta_n},\end{aligned} \tag{34}$$

$$\dot{\mu}_n^{r,j} = \nabla_{\theta^{r,j}}\mu_n^r|_{\theta=\theta_n} = \left(M_n^r\right)^{-1}\dot{z}_n^{r,j}. \tag{35}$$

Using (5) we note that for the set of vertices $i$ for which the path from $r$ to $i$ includes edge $(r,j)$, $\nabla_{\theta^{r,j}}\theta^{r,i} = I$ (the identity matrix) whereas for the rest $\nabla_{\theta^{r,j}}\theta^{r,i} = 0$. For all the nodes $i$ for which $\nabla_{\theta^{r,j}}\theta^{r,i} = I$, let them form a sub tree $(\mathcal{V}_{rj}', \mathcal{E}_{rj}')$ branching out from node $j$ away from node $r$. Then the last sum in the expression for $\nabla_{\theta^{r,j}} \log p_\theta^r(Y_n \,|\, Y_{1:n-1})|_{\theta=\theta_n}$ evaluates to

$$\sum_{i\in\mathcal{V}_{rj}'}\left(C_n^i\right)^{\mathrm{T}}\left(R_n^i\right)^{-1}\left(Y_n^i - C_n^i\theta_n^{r,i}\right) = \dot{m}_{n,K}^{j,r} - \ddot{m}_{n,K}^{j,r}$$

where messages $(\dot{m}_{n,K}^{j,r}, \ddot{m}_{n,K}^{j,r})$ were defined in Algorithm 2. Similarly, we can write the sum in the expression for $\dot{z}_n^{r,j}$ as $m_{n,K}^{j,r}$ (again refer to Algorithms 2) to obtain

$$\dot{z}_n^{r,j} = \left(\Sigma_{n\,|\,n-1}^r\right)^{-1}\dot{\mu}_{n\,|\,n-1}^{r,j} - m_{n,K}^{j,r}. \tag{36}$$

To conclude, the approximations to $(\dot{\mu}_n^{r,j} = \nabla_{\theta^{r,j}} \mu_n^r|_{\theta=\theta_{n+1}}, \mu_n^r|_{\theta=\theta_{n+1}}, \Sigma_n^r)$ for the subsequent RML iteration, i.e., (25) at time $n = n+1$, are given by

$$\dot{\mu}_n^{r,j} = (M_n^r)^{-1} \dot{z}_n^{r,j}$$

while $(\mu_n^r|_{\theta=\theta_{n+1}}, \Sigma_n^r)$ are given by (19)–(22). The approximation to $\nabla_{\theta^{r,j}} \mu_n^r|_{\theta=\theta_{n+1}}$ follows from differentiating (22). $(\dot{\mu}_n^{r,j}, \mu_n^r|_{\theta=\theta_{n+1}})$ are only approximations because they are computed using the previous values of the parameters, i.e., $\theta_{1:n}$.

## Appendix B
## Distributed EM Derivation

For the off-line EM approach, once a batch of $T$ observations have been obtained, each node $r$ of the network that controls an edge will execute the following E and M step iteration $n$,

$$Q^r(\theta_p, \theta) = \int \log p_\theta^r(x_{1:T}^r, Y_{1:T})\, p_{\theta_p}^r(x_{1:T}^r \mid Y_{1:T})\, dx_{1:T}^r,$$
$$\theta_{p+1}^{r,j} = \arg\max_{\theta^{r,j} \in \Theta} Q^r(\theta_p, (\theta^{r,j}, \{\theta^e, e \in \mathcal{E} \backslash (r,j)\}))$$

where it is assumed that node $r$ controls edge $(r,j)$. The quantity $p_{\theta_p}^r(x_{1:T}^r \mid Y_{1:T})$ is the joint distribution of the hidden states at node $r$ given all the observations of the network from time 1 to $T$ and is given up to a proportionality constant

$$p_{\theta_p}^r(x_{1:T}^r)\, p_{\theta_p}^r(Y_{1:T} \mid x_{1:T}^r) = \prod_{n=1}^T f_n(x_n^r \mid x_{n-1}^r)\, p_{\theta_p}^r(Y_n \mid x_n^r)$$

where $p_{\theta_p}^r(Y_n \mid x_n^r)$ was defined in (9). Note that $p_{\theta_p}^r(x_{1:T}^r, Y_{1:T})$ (and hence $p_{\theta_p}^r(x_{1:T}^r \mid Y_{1:T})$) is a function of $\theta_p = \{\theta_p^{i,i'}\}_{(i,i') \in \mathcal{E}}$ and not just $\theta_p^{r,j}$. Also, the $\theta$-dependence of $p_\theta^r(x_{1:T}^r, Y_{1:T})$ arises through the likelihood term only as $p_\theta^r(x_{1:T}^r)$ is $\theta$ independent. Note that

$$\begin{aligned}\sum_{v \in \mathcal{V}} & \log g_n^v(Y_n^v \mid x_n^r + \theta^{r,v}) \\ &= \sum_{v\in\mathcal{V}} c_n^v - \frac{1}{2}\sum_{v\in\mathcal{V}} (Y_n^v - C_n^v\theta^{r,v})^{\mathrm{T}} (R_n^v)^{-1} (Y_n^v - C_n^v\theta^{r,v}) \\ &\quad + (x_n^r)^{\mathrm{T}} \sum_{v\in\mathcal{V}} (C_n^v)^{\mathrm{T}} (R_n^v)^{-1} (Y_n^v - C_n^v\theta^{r,v}) \\ &\quad - \frac{1}{2}(x_n^r)^{\mathrm{T}} \left[\sum_{v\in\mathcal{V}} (C_n^v)^{\mathrm{T}} (R_n^v)^{-1} C_n^v\right] x_n^r\end{aligned}$$

where $c_n^v$ is a constant independent of $\theta$. Taking the expectation w.r.t. $p_{\theta_n}^r(x_n^r \mid Y_{1:T})$ gives

$$\begin{aligned}\int & \log p_\theta^r(Y_n \mid x_n^r)\, p_{\theta_p}^r(x_n^r \mid Y_{1:T})\, dx_n^r \\ &= -\frac{1}{2}\sum_{v\in\mathcal{V}} \left[(Y_n^v - C_n^v\theta^{r,v})^{\mathrm{T}} (R_n^v)^{-1} (Y_n^v - C_n^v\theta^{r,v})\right] \\ &\quad - \left(\mu_{n\mid T}^r\right)^{\mathrm{T}} \sum_{v\in\mathcal{V}} (C_n^v)^{\mathrm{T}} (R_n^v)^{-1} C_n^v \theta^{r,v} + \text{const}\end{aligned}$$

where all terms independent of $\theta^{r,j}$ have been lumped together as 'const' and $\mu_{n\mid T}^r$ is the mean of $x_n^r$ under $p_{\theta_p}^r(x_n^r \mid Y_{1:T})$. Taking the gradient w.r.t. $\theta^{r,j}$ and following the steps in the derivation of the distributed RML we obtain

$$\begin{aligned}\nabla_{\theta^{r,j}} & \int \log p_\theta^r(Y_n \mid x_n^r)\, p_{\theta_p}^r(x_n^r \mid Y_{1:T})\, dx_n^r \\ &= \dot{m}_{n,K}^{j,r} - \ddot{m}_{n,K}^{j,r} - \left(m_{n,K}^{j,r}\right)^{\mathrm{T}} \mu_{n\mid T}^r\end{aligned}$$

where $(m_{n,K}^{j,r}, \dot{m}_{n,K}^{j,r}, \ddot{m}_{n,K}^{j,r})$ is defined in (16)–(18). Only $\ddot{m}_{n,K}^{j,r}$ is a function of $\theta^{r,j}$. It is trivial then to show that the M-step is given by (29), whereby $\theta^{r,j}$ can be recovered by standard linear algebra using quantities available locally to node $r$ and $j$. One can use the fact that $\sum_{j' \in \mathrm{ne}(j)\backslash\{r\}} \ddot{m}_{n,K-1}^{j',j} = \ddot{m}_{n,K}^{j,r} - \ddot{m}_{n,1}^{j,r}$ to show that the M-step can be performed with quantities available locally to node $r$ only. Recall that $\sum_{n=1}^T \mu_{n\mid T}^r = \int(\sum_{n=1}^T x_n^r) p_{\theta_p}^r(x_{1:T}^r \mid Y_{1:T})\, dx_{1:T}^r$. Therefore the summary statistics should be defined using functions $s_{n,1}^{r,j}$, $s_{n,2}^{r,j}$ and $s_{n,3}^{r,j}$ as shown in (30)–(31) and are given by

$$\begin{aligned}\mathcal{S}_{T,1}^{r,j^{,\theta_p}} &= \frac{1}{T}\int \left(\sum_{n=1}^T \left(m_{n,K}^{j,r}\right)^{\mathrm{T}} x_n^r\right) p_{\theta_p}^r(x_{1:T}^r \mid Y_{1:T})\, dx_{1:T}^r, \\ \mathcal{S}_{T,2}^{r,j^{,\theta_p}} &= \frac{1}{T}\sum_{n=1}^T m_{n,K}^{j,r}, \\ \mathcal{S}_{T,3}^{r,j^{,\theta_p}} &= \frac{1}{T}\sum_{n=1}^T \left(\dot{m}_{n,K}^{j,r} - \ddot{m}_{n,K}^{j,r} + \ddot{m}_{n,1}^{j,r}\right)\end{aligned}$$

and the M-step function becomes $\Lambda(\mathcal{S}_{T,1}^{r,j^{,\theta_p}}, \mathcal{S}_{T,2}^{r,j^{,\theta_p}}, \mathcal{S}_{T,3}^{r,j^{,\theta_p}}) = (\mathcal{S}_{T,2}^{r,j^{,\theta_p}})^{-1}(\mathcal{S}_{T,3}^{r,j^{,\theta_p}} - \mathcal{S}_{T,1}^{r,j^{,\theta_p}})$.

We will now proceed to the on-line implementation. Let at time $n$ the estimate of the localization parameter be $\theta_n$. For every $r \in \mathcal{V}$ and $(r,j) \in \mathcal{E}$, let $\mathcal{S}_{n,1}^{r,j}$, $\mathcal{S}_{n,2}^{r,j}$, $\mathcal{S}_{n,3}^{r,j}$ be the running averages (w.r.t. $n$) for $\mathcal{S}_{T,1}^{r,j^{\theta_p}}$, $\mathcal{S}_{T,2}^{r,j^{\theta_p}}$ and $\mathcal{S}_{T,3}^{r,j^{\theta_p}}$, respectively. The recursions for $\mathcal{S}_{n,2}^{r,j}$, $\mathcal{S}_{n,3}^{r,j}$ are trivial:

$$\begin{aligned}\mathcal{S}_{n,2}^{r,j} &= \gamma_n^r m_{n,K}^{j,r} + (1-\gamma_n^r)\mathcal{S}_{n-1,2}^{r,j}, \\ \mathcal{S}_{n,3}^{r,j} &= \gamma_n^r\left(\dot{m}_{n,K}^{j,r} - \ddot{m}_{n,K}^{j,r} + \ddot{m}_{n,1}^{j,r}\right) + (1-\gamma_n^r)\mathcal{S}_{n-1,3}^{r,j},\end{aligned}$$

where $\{\gamma_n^r\}_{n\geq1}$ needs to satisfy $\sum_{n\geq1}\gamma_n^r = \infty$ and $\sum_{n\geq1}(\gamma_n^r)^2 < \infty$. For $\mathcal{S}_{n,1}^{r,j}$ we first set $V_0^{r,j}(x_0^r) = 0$ and define the recursion

$$\begin{aligned}V_n^{r,j}(x_n^r) &= \gamma_n^r\left(m_{n,K}^{j,r}\right)^{\mathrm{T}} x_n^r + (1-\gamma_n^r) \\ &\quad \times \int V_{n-1}^{r,j}(x_{n-1}^r)\, p_{\theta_{1:n}}^r\left(x_{n-1}^r \,\middle|\, Y_{1:n-1}, x_n^r\right) dx_{n-1}^r. \quad (37)\end{aligned}$$

Using standard manipulations with Gaussians we can derive that $p_{\theta_{1:n}}^r(x_{n-1}^r \mid Y_{1:n-1}, x_n^r)$ is itself a Gaussian density with mean and variance denoted by $\tilde{\mu}_n^r(x_n)$, $\tilde{\Sigma}_n^r$, respectively, where

$$\begin{aligned}\tilde{\Sigma}_n^r &= \left(\Sigma_{n-1}^r + A_n^{\mathrm{T}} Q_n^{-1} A_n\right)^{-1}, \\ \tilde{\mu}_n^r(x_n) &= \tilde{\Sigma}_n^r\left(\left(\Sigma_{n-1}^r\right)^{-1}\mu_{n-1}^r + A_n^{\mathrm{T}} Q_n^{-1} x_n\right).\end{aligned}$$

It is then evident that (37) becomes $V_n^{r,j}(x_n^r) = H_n^{r,j} x_n^r + h_n^{r,j}$, with:

$$H_n^{r,j} = \gamma_n^r \left(m_{n,K}^{j,r}\right)^{\mathrm{T}} + (1-\gamma_n^r) H_{n-1}^{r,j} \left(\tilde{\Sigma}_n^r\right)^{-1} A_n^{\mathrm{T}} Q_n^{-1},$$
$$h_n^{r,j} = (1-\gamma_n^r)\left(H_{n-1}^{r,j}\left(\tilde{\Sigma}_n^r\right)^{-1}\left(\Sigma_{n-1}^r\right)^{-1}\mu_{n-1}^r + h_{n-1}^{r,j}\right)$$

where $H_0^{r,j} = 0$ and $h_0^{r,j} = 0$. Finally, the recursive calculation of $\mathcal{S}_{n,1}^{r,j}$ is achieved by computing

$$\mathcal{S}_{n,1}^{r,j} = \int V_n^{r,j}\left(x_n^r\right) p_{\theta_{0:n}}^r\left(x_n^r \mid Y_{0:n}\right) dx_n^r = H_n^{r,j}\mu_n^r + h_n^{r,j}.$$

Again all the steps are performed locally at node $r$, which can update parameter $\theta^{r,j}$ using $\theta_{n+1}^{r,j} = \Lambda(\mathcal{S}_{n,1}^{r,j}, \mathcal{S}_{n,2}^{r,j}, \mathcal{S}_{n,3}^{r,j})$.

## REFERENCES

[1] N. Patwari, J. N. Ash, S. Kyperountas, A. O. Hero, III, R. L. Moses, and N. S. Correal, "Locating the nodes: Cooperative localization in wireless sensor networks," *IEEE Signal Process. Mag.*, vol. 22, no. 4, pp. 54–69, Jul. 2005.

[2] K. Plarre and P. Kumar, "Tracking objects with networked scattered directional sensors," *EURASIP J. Adv. Signal Process.*, vol. 2008, p. 74, 2008.

[3] N. B. Priyantha, H. Balakrishnan, E. D. Demaine, and S. Teller, "Mobile-assisted localization in wireless sensor networks," in *Proc. 24th Ann. Joint Conf. IEEE Comput. Commun. Soc. INFOCOM*, Mar. 13–17, 2005, vol. 1, pp. 172–183.

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

[5] R. L. Moses, D. Krishnamurthy, and R. Patterson, "A self-localization method for wireless sensor networks," *EURASIP J. Appl. Signal Process. Special Issue on Sens. Netw.*, vol. 2003, no. 4, pp. 348–358, Mar. 2003.

[6] M. Vemula, M. F. Bugallo, and P. M. Djuric, "Sensor self-localization with beacon position uncertainty," *Signal Process.*, pp. 1144–1154, 2009.

[7] V. Cevher and J. H. McClellan, "Acoustic node calibration using moving sources," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 42, no. 2, pp. 585–600, 2006.

[8] X. Chen, A. Edelstein, Y. Li, M. Coates, M. Rabbat, and A. Men, "Sequential Monte Carlo for simultaneous passive device-free tracking and sensor localization using received signal strength measurements," in *Proc. IEEE/ACM Int. Conf. on Inf. Process. Sens. Netw.*, Chicago, IL, 2011.

[9] S. Funiak, C. Guestrin, M. Paskin, and R. Sukthankar, "Distributed localization of networked cameras," in *Proc. 5th Int. Conf. Inf. Process. Sens. Netw. IPSN 2006*, 2006, pp. 34–42.

[10] C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe, and A. Grue, "Simultaneous localization, calibration, and tracking in an ad hoc sensor network," in *Proc. 5th Int. Conf. Inf. Process. Sens. Netw. IPSN* , 2006, pp. 27–33.

[11] Y. Baryshnikov and J. Tan, "Localization for anchoritic sensor networks," in *Proc. 3rd IEEE Int. Conf. Distrib. Comput. Sens. Syst. (DCOSS'07)*, Santa Fe, NM, Jun. 18–20, 2007.

[12] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. New York: Morgan-Kauffman, 1988.

[13] N. Kantas, "Stochastic Decision Making for General State Space Models," Ph.D. dissertation, Univ. Cambridge, Cambridge, U.K., Feb. 2009.

[14] O. Cappé, E. Moulines, and T. Rydén, *Inference in Hidden Markov Models*. New York: Springer, 2005.

[15] R. Elliott, L. Aggoun, and J. Moore, *Hidden Markov Models: Estimation and Control*. New York: Springer-Verlag, 1995.

[16] U. Holst and G. Lindgren, "Recursive estimation in mixture models with Markov regime," *IEEE Trans. Inf. Theory*, vol. 37, no. 6, pp. 1683–1690, Nov. 1991.

[17] F. LeGland and L. Mevel, "Recursive estimation in hidden Markov models," in *Proc. 36th IEEE Conf. Decision Contr.*, Dec. 10–12, 1997, vol. 4, pp. 3468–3473.

[18] I. B. Collings and T. Ryden, "A new maximum likelihood gradient algorithm for on-line hidden Markov model identification," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 12–15, 1998, vol. 4, pp. 2261–2264.

[19] J. J. Ford, "Adaptive Hidden Markov Model Estimation and Applications," Ph.D. dissertation, Dep. Syst. Eng., Austral. Nat. Univ., , 1998.

[20] R. Elliott, J. Ford, and J. Moore, On-Line Consistent Estimation of Hidden Markov Models Dep. Syst. Eng., Austral. Nat. Univ., Tech. Rep., 2000.

[21] O. Cappe and E. Moulines, "Online em algorithm for latent data models," *J. Royal Statist. Soc. Ser. B (Methodolog.)*, vol. 71, p. 593, 2009.

[22] *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. de Freitas, and N. Gordon, Eds. New York: Springer, 2001.

[23] S. Grime and H. Durrant-Whyte, "Data fusion in decentralized sensor networks," *Contr. Eng. Practice*, vol. 2, no. 1, pp. 849–863, 1994.

[24] O. Cappe, "Online em algorithm for hidden Markov models," *J. Computat. Graph. Statist.*, 2011.

[25] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proc. 44th Ann. IEEE Symp. Found. Comput. Sci.*, 2003, pp. 482–491.

[26] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation*. Old Tappan, NJ: Prentice-Hall, 1989.

**Nikolas Kantas** was born in Athens, Greece, in 1981. He received the B.A., M.Eng., and Ph.D. degrees from Cambridge University, Cambridge, U.K., in 2003 and 2009, respectively.

He has since held appointments as a Research Associate in the Engineering Department, Cambridge University (2008–2010) and in the Department of Electrical and Electronic Engineering, Imperial College London, U.K. (2010–present). His research interests are Monte Carlo methods for inference and optimization problems.

**Sumeetpal S. Singh** received the B.E. (with first-class honours) and Ph.D. degrees from the University of Melbourne, Melbourne, Australia, in 1997 and 2002, respectively.

From 1998 to 1999, he was a Communications Engineer in industry. From 2002 to 2004, he was a Research Fellow with the Department of Electrical and Electronic Engineering, University of Melbourne. He joined the Engineering Department of Cambridge University, Cambridge, U.K., in 2004 as a Research Associate and is currently a University Lecturer in Engineering statistics. He is also a Fellow of Churchill College and an Affiliated Lecturer at the Statistics Laboratory of Cambridge University. His research focus is on statistical signal processing, in particular, Monte Carlo methods for estimation and control.

**Arnaud Doucet** was born in France in 1970. He received the Ph.D. degree in information engineering from University Paris XI (Orsay), France, in 1997.

He has previously held academic appointments at the University of Melbourne, the University of Cambridge, the University of British Columbia, and the Institute of Statistical Mathematics. Since 2011, he is Professor with the Department of Statistics of the University of Oxford, Kay Fellow of Statistics of Jesus College, and a member of the Oxford-Man Institute. His main research interests are Bayesian statistics and Monte Carlo methods.