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

N. Kantas is with the Control and Power Group, Department of Electical and Electronic Engineering, Imperial College, London, UK, SW7 2AZ, e-mail: {n.kantas@imperial.ac.uk}.

S.S. Singh is with the Signal Processing lab, Department of Engineering, University of Cambridge, Trumpington Road, Cambridge, UK, CB2 1PZ, e-mail: {sss40@cam.ac.uk}.

A. Doucet is with the Department of Statistics, University of Oxford, 1 South Parks Road, Oxford, OX1 3TG, e-mail: doucet@stats.ox.ac.uk

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.

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], [2], [3], [4], [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], [3], [4], [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], [7], [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], [10], [11] seems to have been overlooked in the localization literature, is the basis of our solution[1]. 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. [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 [13] for details) and requires an anchor node as pointed out in [14, Section 6.2.3]. In this paper we formulate the sensor localization problem as a static parameter estimation problem for Hidden Markov Models (HMMs) [15], [16] 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) [17], [18], [19] and on-line Expectation-Maximization (EM) [20], [21], [22]. 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 centralized in nature.

In the signal processing literature for sensor networks one may find various related problems. In [23] a distributed EM algorithm was developed to estimate the parameters of a Gaussian mixture used to model the measurements of a sensor network deployed for environmental monitoring (see [24] for an on-line version.) In [25] a similar problem is treated using a distributed gradient method. We emphasize that in each of these papers the measurements correspond to a static source instead of a dynamically evolving target. In addition, a related problem is that of *sensor registration,* which aims to compensate for systematic biases in the sensors and has been studied by the target tracking community [26], [27]. However, the algorithms devised in [26], [27] are centralized. Yet another related problem is the problem

[1]A short preliminary version of the this work was published in the conference proceedings [12].

of average consensus [28]. The value of a global static parameter is measured at each node via a linear Gaussian observation model and the aim is to obtain a maximum likelihood estimate in a distributed fashion. Note that all the aforementioned papers, except [9] and [10], do not deal with a distributed localization and tracking task.

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):

$$X_n^r | X_{n-1}^r = x_{n-1}^r \sim f_n(.|x_{n-1}^r), \tag{3}$$

$$Y_n^r | X_n^r = x_n^r \sim g_n^r(.|x_n^r), \tag{4}$$

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

Figure 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 Figure 1):

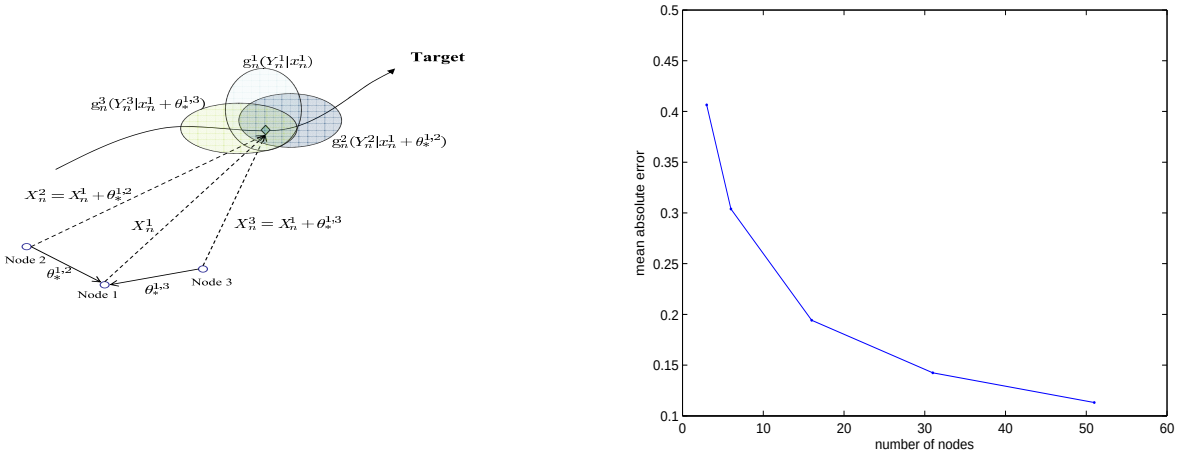
$$X_n^j = X_n^i + \theta_*^{i,j}.$$

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

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

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

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



(a) Three node joint tracking example

(b) Joint tracking error vs number of nodes

Figure 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 $g_n^1(Y_n^1|.)$ 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 $A_n = B_n = Q_n = C_n^i = D_n^i = R_n^i = 1$, $b_n^i = d_n^i = 0$.

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 $(Y_1, ..., Y_n)$ by $Y_{1:n}$. In the *collaborative or joint* filtering problem, each node r computes the local filtering density:

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

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

$$p_{\theta_*}^r(x_n^r|Y_{1:n-1}) = \int f_n(x_n^r|x_{n-1}^r)p_{\theta_*}^r(x_{n-1}^r|Y_{1:n-1})dx_{n-1}^r. \tag{8}$$

The likelihood term is

$$p^r_{\theta_*}(Y_n|x^r_n) = \prod_{v\in\mathcal{V}} g^v_n(Y^v_n|x^r_n + \theta^{r,v}_*), \tag{9}$$

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 $\mu^r_{n|n-1}$ and μ^r_n denote the predicted and filtered mean of the densities $p^r_{\theta_*}(x^r_n|Y_{1:n-1})$ and $p^r_{\theta_*}(x^r_n|Y_{1:n})$ respectively, where the dependence on θ_* 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. Figure 1 (a) shows the support of the three observation densities as ellipses where the support of $g^1_n(Y^1_n|\cdot)$ is defined to be all x^1 such that $g^1_n(Y^1_n|\cdot) > 0$; 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 $p^1_{\theta_*}(Y_n|x^1_n)$ defined in (9) is

$$g^1_n(Y^1_n|x^1_n)g^2_n(Y^2_n|x^1_n + \theta^{1,2}_*)g^3_n(Y^3_n|x^1_n + \theta^{1,3}_*).$$

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 Figure 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 θ_* 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. Non-linear Model

Most tracking problems of practical interest are essentially non-linear non-Gaussian filtering problems. SMC methods, also known as Particle Filters, provide very good approximations to the filtering densities [29]. 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 [14, Chapter 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:

$$X_n^r = \phi_n(X_{n-1}^r) + V_n, \tag{10}$$

$$Y_n^r = \psi_n^r(X_n^r) + W_n^r, \tag{11}$$

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:

$$X_n^r = \phi_n(\mu_{n-1}^r) + \nabla\phi_n(\mu_{n-1}^r)(X_{n-1}^r - \mu_{n-1}^r) + V_n, \tag{12}$$

$$Y_n^r = \psi_n^r(\mu_{n|n-1}^r) + \nabla\psi_n^r(\mu_{n|n-1}^r)(X_n^r - \mu_{n|n-1}^r) + W_n^r. \tag{13}$$

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 equations (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 θ_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. It would require a prohibitive amount of communication even for relatively few nodes and redundant computations would be performed at the different nodes. The core idea in this paper is to avoid this by storing the parameters in θ_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.

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

Algorithm 1 Generic message passing at time n

1: **begin**

2: At $k = 1$, compute:

$$m_{n,1}^{i,j} = F_n^i, \tag{14}$$

$$\ddot{m}_{n,1}^{i,j} = F_n^i \theta_n^{j,i}. \tag{15}$$

3: **for** $k = 2, ..., K$ compute:

$$m_{n,k}^{i,j} = F_n^i + \sum_{p \in \text{ne}(i) \setminus \{j\}} m_{n,k-1}^{p,i}, \tag{16}$$

$$\ddot{m}_{n,k}^{i,j} = m_{n,k}^{i,j} \theta_n^{j,i} + \sum_{p \in \text{ne}(i) \setminus \{j\}} \ddot{m}_{n,k-1}^{p,i}. \tag{17}$$

4: **endfor**

5: **end**

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 $\sum_{v \in \mathcal{V}} F_n^v = F_n^r + \sum_{j \in \text{ne}(r)} m_{n,K}^{j,r}$ and $\sum_{v \in \mathcal{V}} F_n^v \theta_n^{r,v} = \sum_{j \in \text{ne}(r)} \ddot{m}_{n,K}^{j,r}$.

(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 $p_\theta^r(x_n^r | Y_{1:n})$ at node r is a Gaussian distribution with a specific mean vector μ_n^r and covariance matrix Σ_n^r. The derivation of the Kalman filter to implement $p_\theta^r(x_n^r | Y_{1:n})$ is standard upon noting that the measurement model at node r can be written as $Y_n =$

$C_nX_n^r+d_n+W_n$ 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 (18)-(20) 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 [30]. However, the message passing scheme is different and due to the application in mind we have extra terms relevant to the localization parameters.

In the case $b_n^i, d_n^i \neq 0$ modifications to Algorithm 2 are as follows: in (21), to the right hand side of $\mu_{n|n-1}^r$, 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 $\mu_{n|n-1}^r$ in (21), and replacing all instances of Y_n^r with $Y_n^r - \psi_n^r(\mu_{n|n-1}^r) + \nabla\psi_n^r(\mu_{n|n-1}^r)\mu_{n|n-1}^r$. 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). For the sake of completeness, we have added brief descriptions of these techniques in Section A of the appendix.

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 of $\theta_*^{r,j}$, which is the component of θ_* corresponding to edge (r,j). This can be achieved computing at each node r the ML

Algorithm 2 Distributed Filtering

begin

for $n \geq 1$:

Let the localization parameter be θ_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:

$$m_{n,1}^{i,j} = (C_n^i)^{\mathrm{T}}(R_n^i)^{-1}C_n^i,$$
$$\dot{m}_{n,1}^{i,j} = (C_n^i)^{\mathrm{T}}(R_n^i)^{-1}Y_n^i,$$
$$\ddot{m}_{n,1}^{i,j} = m_n^{i,j}\theta_n^{j,i},$$

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 $(i,j) \in \mathcal{E}$:

$$m_{n,k}^{i,j} = (C_n^i)^{\mathrm{T}}(R_n^i)^{-1}C_n^i + \sum_{p\in\mathrm{ne}(i)\setminus\{j\}} m_{n,k-1}^{p,i}, \tag{18}$$

$$\dot{m}_{n,k}^{i,j} = (C_n^i)^{\mathrm{T}}(R_n^i)^{-1}Y_n^i + \sum_{p\in\mathrm{ne}(i)\setminus\{j\}} \dot{m}_{n,k-1}^{p,i}, \tag{19}$$

$$\ddot{m}_{n,k}^{i,j} = m_n^{i,j}\theta_n^{j,i} + \sum_{p\in\mathrm{ne}(i)\setminus\{j\}} \ddot{m}_{n,k-1}^{p,i}, \tag{20}$$

end for

Update the local filtering densities at each node $r \in \mathcal{V}$:

$$\mu_{n|n-1}^r = A_n\mu_{n-1}^r, \quad \Sigma_{n|n-1}^r = A_n\Sigma_{n-1}^r A_n^{\mathrm{T}} + Q_n, \tag{21}$$

$$M_n^r = (\Sigma_{n|n-1}^r)^{-1} + (C_n^r)^{\mathrm{T}}(R_n^r)^{-1}C_n^r + \sum_{i\in\mathrm{ne}(r)} m_n^{i,r} \tag{22}$$

$$z_n^r = (\Sigma_{n|n-1}^r)^{-1}\mu_{n|n-1}^r + (C_n^r)^{\mathrm{T}}(R_n^r)^{-1}Y_n^r + \sum_{i\in\mathrm{ne}(r)} \left(\dot{m}_n^{i,r} - \ddot{m}_n^{i,r}\right), \tag{23}$$

$$\Sigma_n^r = (M_n^r)^{-1}, \quad \mu_n^r = \Sigma_n^r z_n^r, \tag{24}$$

end for

end

estimate:

$$\widetilde{\theta}_n^{r,j} = \arg \max_{\theta^{r,j} \in \mathbb{R}^4} \log p_\theta^r(Y_{1:n}). \tag{25}$$

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

$$\theta_{n+1}^{r,j} = G_{n+1}^{r,j}(\theta_n, Y_n), \qquad n \geq 1, \tag{26}$$

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 $\theta_n^{j,r}$. While updating both parameters associated to each edge is redundant, it allows a fully decentralized implementation since no other communication is needed other than the messages defined in Algorithm 1. Alternatively one could assign both parameters of an edge to just one controlling node. For example in the three node network of Figure 1, the parameters of edge $(1, 2)$, $\theta_n^{1,2}$ and $\theta_n^{2,1}$, could be assigned to node 2, with the latter having at each time n to update $\theta_n^{2,1}$ using an expression like (26) and then send $\theta_n^{1,2} = -\theta_n^{2,1}$ to node 1.

A. Distributed RML

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

$$\theta_{n+1}^{r,j} = \theta_n^{r,j} + \gamma_{n+1}^r \left[\nabla_{\theta^{r,j}} \log \int p_\theta^r(Y_n|x_n^r) p_\theta^r(x_n^r|Y_{1:n-1}) dx_n^r \right]_{\theta=\theta_n}, \tag{27}$$

where γ_{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 (27) is w.r.t. $\theta^{r,j}$. The local joint *predicted* density $p_\theta^r(x_n^r|Y_{1:n-1})$ 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 θ_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 (27) in the present form is not an on-line parameter update like (26) as it requires browsing through the entire history of observations.

Algorithm 3 Distributed RML

1: **begin**

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

3: **Filtering step**: Perform steps (3-6) in Algorithm 2.

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

$$\dot{\mu}_{n|n-1}^{r,j} = A_n \dot{\mu}_{n-1}^{r,j}, \tag{28}$$

$$\dot{z}_n^{r,j} = (\Sigma_{n|n-1}^r)^{-1} \dot{\mu}_{n|n-1}^{r,j} - m_{n,K}^{j,r}, \tag{29}$$

$$\dot{\mu}_n^{r,j} = (M_n^r)^{-1} \dot{z}_n^{r,j}. \tag{30}$$

Upon doing so the localization parameter is updated:

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

5: **end for**

6: **end**

This limitation is removed by defining certain intermediate quantities that facilitate the online evaluation of this gradient in the spirit of [18], [19] (see in the Appendix for more details).

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 (28)-(30) 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 θ_* 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 $p = 1, 2, \ldots$

$$\text{(E step) } Q^r(\theta_p, \theta) = \int \log p^r_\theta(x^r_{1:T}, Y_{1:T}) p^r_{\theta_p}(x^r_{1:T}|Y_{1:T}) dx^r_{1:T},$$

$$\text{(M step)} \qquad \theta^{r,j}_{p+1} = \arg\max_{\theta^{r,j}} \quad Q^r(\theta_p, (\theta^{r,j}, \theta_p^{-(r,j)})),$$

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

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

$$p^r_\theta(x^r_{1:T}) p^r_\theta(Y_{1:T}|x^r_{1:T}) = \prod_{n=1}^{T} f_n(x^r_n|x^r_{n-1}) p^r_\theta(Y_n|x^r_n),$$

where $p^r_\theta(Y_n|x^r_n)$ was defined in (9). Note that $p^r_{\theta_p}(x^r_{1:T}|Y_{1:T})$ is a function of $\theta_p = \{\theta_p^{i,i'}\}_{(i,i') \in \mathcal{E}}$ (and not just $\theta^{r,j}_p$) and the θ-dependance of $p^r_\theta(x^r_{1:T}, Y_{1:T})$ arises through the likelihood term only as $p^r_\theta(x^r_{1:T})$ is θ-independent. This means that in order to compute the E-step, it is sufficient to maintain the smoothed marginals:

$$p^r_\theta(x^r_n|Y_{1:T}) \propto \int p^r_\theta(x^r_{1:T}, Y_{1:T}) dx^r_{1:T\setminus\{n\}},$$

where $1 \le n \le T$ and $dx^r_{1:T\setminus\{n\}}$ means integration w.r.t. all variables except x^r_n. For linear Gaussian models this smoothed density is also Gaussian, with its mean and covariance denoted by $\mu^r_{n|T}, \Sigma^r_{n|T}$ 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:

$$\nabla_{\theta^{r,j}} \int \log p^r_\theta(Y_n|x^r_n) p^r_{\theta_p}(x^r_n|Y_{1:T}) dx^r_n = \dot{m}^{j,r}_{n,K} - \ddot{m}^{j,r}_{n,K} - (m^{j,r}_{n,K})^{\mathrm{T}} \mu^r_{n|T},$$

where $(m^{j,r}_{n,K}, \dot{m}^{j,r}_{n,K}, \ddot{m}^{j,r}_{n,K})$, defined in (18)-(20), are propagated with localization parameter θ_p for all observations from time 1 to T. Only $\ddot{m}^{j,r}_{n,K}$ is a function of $\theta^{r,j}$. To perform the M-step, the following equation is solved for $\theta^{r,j}$

$$(\sum_{n=1}^{T} m^{j,r}_{n,K}) \theta^{r,j} = \sum_{n=1}^{T} (\dot{m}^{j,r}_{n,K} - (m^{j,r}_{n,K})^{\mathrm{T}} \mu^r_{n|T} - \ddot{m}^{j,r}_{n,K} + \ddot{m}^{j,r}_{n,1}) \tag{31}$$

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:

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

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

$$\mathcal{S}^{r,j}_{T,l} = \frac{1}{T} \int \left(\sum_{n=1}^{T} s^{r,j}_{n,l}(x^r_n, Y_n) \right) p^r_{\theta_p}(x^r_n|Y_{1:T}) dx^r_n, \; l = 1, 2, 3,$$

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

$$\begin{aligned} s_{n,1}^{r,j}(x_n^r, Y_n) &= (m_{n,K}^{j,r})^{\mathrm{T}} x_n^r, \qquad s_{n,2}^{r,j}(x_n^r, Y_n) = m_{n,K}^{j,r}, \\ s_{n,3}^{r,j}(x_n^r, Y_n) &= \dot{m}_{n,K}^{j,r} - \ddot{m}_{n,K}^{j,r} + \ddot{m}_{n,1}^{j,r}. \end{aligned}$$

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 Λ 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. The derivation of the recursions for $\mathcal{S}_{n,1}^{r,j}$, $\mathcal{S}_{n,2}^{r,j}$, $\mathcal{S}_{n,3}^{r,j}$ are based on (42)-(43) in the Appendix. Here γ_n^r is a step-size satisfying the same conditions as in RML and θ_0 can be initialized arbitrarily, e.g. the zero vector. Finally, it has been reported in [31] 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.

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 $V_n = B\widetilde{V}_n$, where $\widetilde{V}_n$ is zero mean Gaussian additive noise with variance $\widetilde{Q}_n$, and

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

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 α^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:

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

Algorithm 4 Distributed on-line EM

1: **begin**

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

3: **Filtering step**: Perform steps (3-6) in Algorithm 2. Also compute

$$\tilde{\Sigma}_n^r = \left(\Sigma_{n-1}^r + A_n^{\mathrm{T}} Q_n^{-1} A_n\right)^{-1}.$$

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

$$\begin{aligned}
H_n^{r,j} &= \gamma_n^r (m_{n,K}^{j,r})^{\mathrm{T}} + (1-\gamma_n^r) H_{n-1}^{r,j} \left(\tilde{\Sigma}_n^r\right)^{-1} A_n^{\mathrm{T}} Q_n^{-1}, \\
h_n^{r,j} &= (1-\gamma_n^r)\left(H_{n-1}^{r,j}\left(\tilde{\Sigma}_n^r\right)^{-1}\left(\Sigma_{n-1}^r\right)^{-1}\mu_{n-1}^r + h_{n-1}^{r,j}\right), \\
\mathcal{S}_{n,1}^{r,j} &= H_n^{r,j}\mu_n^r + h_n^{r,j}. \\
\mathcal{S}_{n,2}^{r,j} &= \gamma_n^r m_{n,K}^{j,r} + (1-\gamma_n^r)\,\mathcal{S}_{n-1,2}^{r,j}, \\
\mathcal{S}_{n,3}^{r,j} &= \gamma_n^r(\dot{m}_{n,K}^{j,r} - \ddot{m}_{n,K}^{j,r} + \ddot{m}_{n,1}^{j,r}) + (1-\gamma_n^r)\,\mathcal{S}_{n-1,3}^{r,j},
\end{aligned}$$

Upon doing so the localization parameter is updated:

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

5: **end for**

6: **end**

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

In Figure 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 Figure 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 Figures 2(a) and 2(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 Λ of summary statistics.

- in (c) and (f) we use distributed RML and on-line EM for respectively for the network of Figure 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 Figure 3 one can notice that the distributed RML and EM algorithms require comparable amount of time to converge with the RML being usually faster. For example in Figures 3 (a) and (d) we observe that RML requires around 1000 iterations to converge whereas on-line EM requires approximately 2000 iterations. We note that the converge rate also depends on the specific network used, the value of K and the simulation parameters.

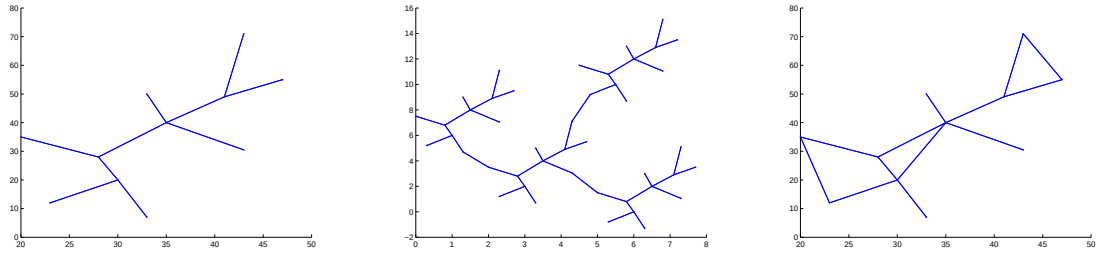
To investigate this further we varied K and $\frac{\sigma_x}{\sigma_y}$ and recorded the root mean squared error (RMSE) for θ_n obtained for the network of Figure 2(b) using 50 independent runs. For the RMSE at time n we will use $\sqrt{\frac{1}{50|\mathcal{E}|}\sum_{e\in\mathcal{E}}\sum_{m=1}^{50}\left\|\theta_*^{r,j} - \theta_{n,m}^{r,j}\right\|_2^2}$, where $\theta_{n,m}^{r,j}$ denotes the estimated parameter at epoch n obtained from the m-th run. The results are plotted in Figure 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 immediate neighboors are not necessary to localise 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.

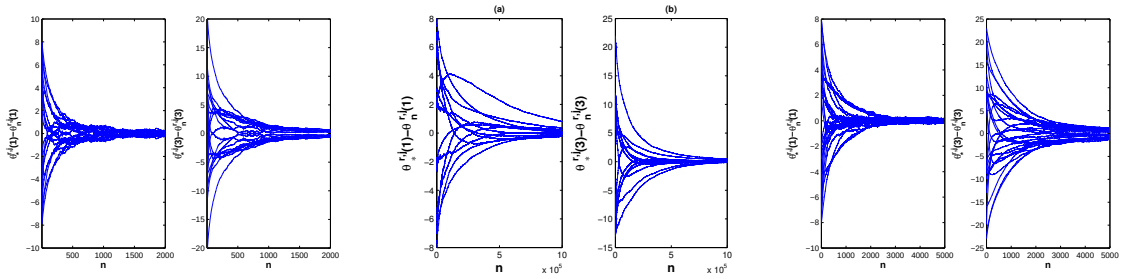
VI. Conclusion

We have presented a method to perform collaborative tracking and self-localization. We exploited the fact that different nodes collect measurements of a common target. This idea has appeared previously in

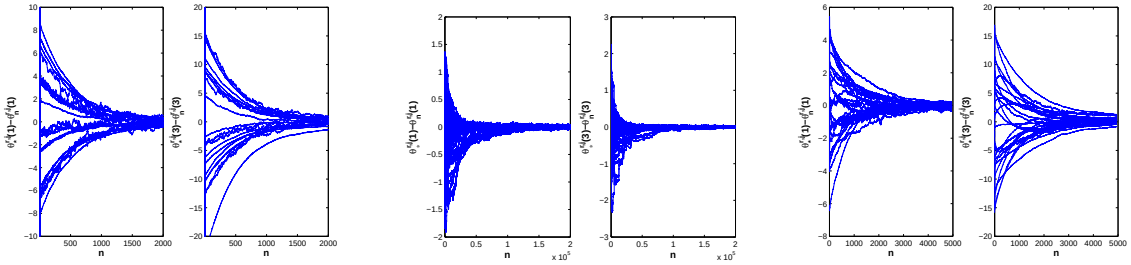


(a) 11 node sensor network (b) 44 node sensor network (c) 11 node sensor network with cycles

Figure 2. Various sensor networks of different size and topology.

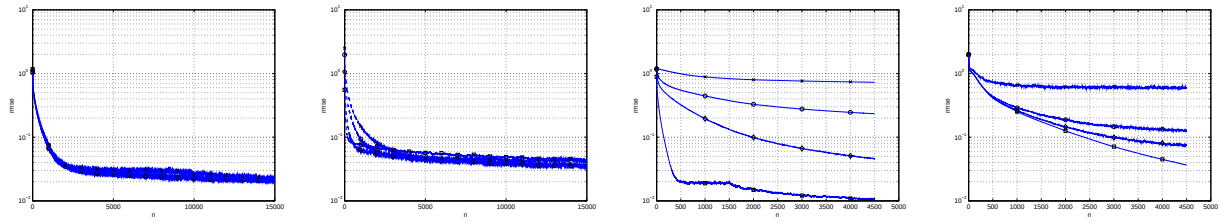


(a) RML for tree network (b) Nonlinear RML for tree network (c) RML for network with cycles



(d) EM for tree network (e) Nonlinear RML for large network (f) EM for network with cycles

Figure 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 different K (b) EM for different K (c) RML for different $\frac{\sigma_x}{\sigma_y}$ (d) EM for different $\frac{\sigma_x}{\sigma_y}$

Figure 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($\times$).

[9], [10], both of which use a Bayesian inference scheme for the localization parameters. We remark that our distributed ML methods appear simpler to implement than these Bayesian schemes as the messages here are nothing more than 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 [32] or the aggregation algorithm of [33] which might be particularly relevant for networks with cycles, which are dealt with here by using an appropriate value for K.

APPENDIX A
MAXIMUM LIKELIHOOD PARAMETER ESTIMATION

This section does not pertain to sensor localization specifically but to the general problem of static parameter estimation in HMMs using ML. Thus to avoid confusion with the localization problem a different font is used the notation. Consider a HMM where $\{\mathsf{X}_n\}_{n\geq 1}$ is the hidden state-process and $\{\mathsf{Y}_n\}_{n\geq 1}$ is the observed process each taking values in taking values in $\mathbb{R}^{d_x}$ and $\mathbb{R}^{d_y}$ respectively. For the transition density for $\{\mathsf{X}_n\}_{n\geq 1}$, we have $\mathsf{X}_{n+1}|\mathsf{X}_n = \mathsf{x}_n \sim \mathsf{f}(\cdot|\mathsf{x}_n)$. The observation model, $\mathsf{Y}_n|\mathsf{X}_n = \mathsf{x}_n \sim \mathsf{g}_\vartheta(\cdot|\mathsf{x}_n)$ is parametrized by $\vartheta \in \Theta$ ($\subset \mathbb{R}^{d_\vartheta}$). The true static parameter generating the sequence of observations is ϑ_* and is to be learned from the observed data $\{\mathsf{Y}_n\}_{n\geq 1}$. The ML parameter estimate is the maximizing argument of the log-likelihood of the observed data up to time n:

$\tilde{\vartheta}_n = \arg\max_{\vartheta\in\Theta} \log p_\vartheta(\mathsf{Y}_{1:n})$. Here $p_\vartheta(\mathsf{Y}_{1:n})$ denotes the joint density of $\mathsf{Y}_{1:n}$ and the subscript makes explicit the value of the parameter used to compute this density.

For a long observation sequence we are interested in a recursive parameter estimation procedure in which the data is run through once sequentially. If ϑ_n is the estimate of the model parameter after n observations, a recursive method would update the estimate to ϑ_{n+1} after receiving the new data Y_n. For example, consider the following update scheme:

$$\vartheta_{n+1} = G_{n+1}(\vartheta_n, \mathsf{Y}_n), \qquad n \geq 1. \tag{33}$$

where G_{n+1} is an appropriate function to be defined. This scheme was originally suggested by [34], [35] when $\{\mathsf{X}_n\}_{n\geq 1}$ is not a Markov chain but rather an independent and identically distributed (i.i.d.) sequence.

A. Recursive Maximum Likelihood (RML)

To motivate a suitable choice for $G_{n+1}(\vartheta_n, \mathsf{Y}_n)$ for estimating the parameters of a HMM, consider the following recursion:

$$\vartheta_{n+1} = \vartheta_n + \gamma_{n+1}\, \nabla \log p_\vartheta(\mathsf{Y}_n|\mathsf{Y}_{1:n-1})|_{\vartheta=\vartheta_n}\,. \tag{34}$$

where $\{\gamma_n\}$ is the step-size sequence that should satisfy the following constraints: $\sum_n \gamma_n = \infty$ and $\sum_n \gamma_n^2 < \infty$. One possible choice would be $\gamma_n = n^{-\alpha}$, $0.5 < \alpha < 1$. Here $p_\vartheta(\mathsf{Y}_n|\mathsf{Y}_{1:n-1})$ is the conditional density of Y_n given $\mathsf{Y}_{1:n-1}$ and the subscript makes explicit the value of the parameter used to compute this density. Upon receiving Y_n, ϑ_n is updated in the direction of ascent of the conditional density of this new observation. The algorithm in the present form is not suitable for online implementation due to the need to evaluate the gradient of $\log p_\vartheta(\mathsf{Y}_n|\mathsf{Y}_{1:n-1})$ (w.r.t. ϑ) at $\vartheta = \vartheta_n$. Doing so would require browsing through the entire history of observations. This limitation is removed by defining certain intermediate quantities that facilitate the online evaluation of this gradient [18], [19].

In particular, assume that from the previous iteration of the RML, one has computed $p_n(\mathsf{x}_n) \approx p_\vartheta(\mathsf{x}_n|\mathsf{Y}_{1:n-1})|_{\vartheta=\vartheta_n}$ and $\dot{p}_n(\mathsf{x}_n) \approx \nabla p_\vartheta(\mathsf{x}_n|\mathsf{Y}_{1:n-1})|_{\vartheta=\vartheta_n}$, where $(p_n, \dot{p}_n)$ are approximations of the predicted density and its gradient evaluated at $\vartheta = \vartheta_n$. The RML is initialized with an arbitrary value for ϑ_1, $p_1(\mathsf{x}_1) = p_{\vartheta_1}(\mathsf{x}_1)$, which is the prior distribution for X_1 and $\dot{p}_1(\mathsf{x}_1) = \nabla p_\vartheta(\mathsf{x}_1)|_{\vartheta=\vartheta_1}$, i.e. the gradient of this prior which could be zero if it does not depend on ϑ. Then the online version of (34), which is the RML procedure of [18], [19], proceeds as follows. Given the new observation Y_n, update

the parameter:

$$\vartheta_{n+1} = \vartheta_n + \gamma_{n+1} \left(\int \mathsf{g}_{\vartheta_n}(\mathsf{Y}_n|\mathsf{x}_n) p_n(\mathsf{x}_n) d\mathsf{x}_n \right)^{-1} \left(\int \dot{\mathsf{g}}_{\vartheta_n}(\mathsf{Y}_n|\mathsf{x}_n) p_n(\mathsf{x}_n) d\mathsf{x}_n + \int \mathsf{g}_{\vartheta_n}(\mathsf{Y}_n|\mathsf{x}_n) \dot{p}_n(\mathsf{x}_n) d\mathsf{x}_n \right) \tag{35}$$

where $n \geq 1$ and $\dot{\mathsf{g}}_{\vartheta'}(\mathsf{y}|\mathsf{x}) \equiv \nabla_\vartheta \mathsf{g}_\vartheta(\mathsf{y}|\mathsf{x})|_{\vartheta=\vartheta'}$. In (34), the desired gradient is the ratio of the terms $p_\vartheta(\mathsf{Y}_n|\mathsf{Y}_{1:n-1})|_{\vartheta=\vartheta_n}$ and $\nabla p_\vartheta(\mathsf{Y}_n|\mathsf{Y}_{1:n-1})|_{\vartheta=\vartheta_n}$. This ratio is approximated in the fraction on the right-hand side of (35). After computing (35), one may update $(p_n, \dot{p}_n)$ to $(p_{n+1}, \dot{p}_{n+1})$ for the next RML iteration. Specific expressions for this update may be found for example in [14, Section 8.2.1] or [18]. The recursive propagation of $(p_n, \dot{p}_n)$ implicitly involves the previous values of the parameter, i.e. $\vartheta_{1:n}$, and hence are only approximations to $p_\vartheta(\mathsf{x}_n|\mathsf{Y}_{1:n-1})|_{\vartheta=\vartheta_{n+1}}$, $\nabla p_\vartheta(\mathsf{x}_n|\mathsf{Y}_{1:n-1})|_{\vartheta=\vartheta_n}$ respectively. It has been shown in [18] that the solution of RML converges to the true ML estimator without any loss of efficiency. For more details on the convergence of RML for HMMs we refer the reader to [18].

B. On-line Expectation-Maximization (EM)

We begin this section with a brief description of Expectation-Maximization (EM) [36] and then present its on-line implementation. EM is an iterative off-line algorithm for learning ϑ_*, which consists of repeating a two step procedure given a batch of T observations. Let p be the (off-line) iteration index. The first step, the expectation or E-step, computes

$$Q(\vartheta_p, \vartheta) = \int \log p_\vartheta(\mathsf{x}_{1:T}, \mathsf{Y}_{1:T}) p_{\vartheta_p}(\mathsf{x}_{1:T}|\mathsf{Y}_{1:T}) d\mathsf{x}_{1:T}. \tag{36}$$

The second step is the maximization or M-step that updates the parameter ϑ_p,

$$\vartheta_{p+1} = \arg\max \quad Q(\vartheta_p, \vartheta) \tag{37}$$

Upon the completion of an E and M step, the likelihood surface is ascended, i.e. $p_{\vartheta_{p+1}}(\mathsf{Y}_{1:T}) \geq p_{\vartheta_p}(\mathsf{Y}_{1:T})$ [36]. When $p_\theta(\mathsf{x}_{1:T}, \mathsf{Y}_{1:T})$ is in the exponential family, which is the case of linear Gaussian state-space models, this procedure can be implemented exactly. Then the E-step is equivalent to computing a summary statistic of the form

$$\mathcal{S}_T^{\vartheta_p} = \frac{1}{T} \int \left(\sum_{n=1}^{T} s_n\left(\mathsf{x}_{n-1:n}, \mathsf{Y}_n\right) \right) p_{\vartheta_p}(\mathsf{x}_{1:T}|\mathsf{Y}_{1:T}) d\mathsf{x}_{1:T}. \tag{38}$$

where $s_n : \mathbb{R}^{d_x} \times \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \to \mathbb{R}^\kappa$. In addition, the maximizing argument of $Q(\vartheta_p, \vartheta)$ can be characterized in this case explicitly through a suitable function $\Lambda : \mathbb{R}^\kappa \to \Theta$, i.e.

$$\vartheta_p = \Lambda\left(\mathcal{S}_T^{\vartheta_p}\right). \tag{39}$$

Note that in the usual EM setup one has to compute (38) for every iteration p of the algorithm.

It is also possible to propose an on-line version of the EM algorithm. This was originally proposed for finite state-space and linear Gaussian models in [21], [37], [20] and for exponential family models in [22], [31]. In the online implementation of the EM, running averages of the sufficient statistics are computed [20], [21], [22]. Let $\{\vartheta_m\}_{1\leq m\leq n}$ be the sequence of parameter estimates of the online EM algorithm computed sequentially based on $\mathsf{Y}_{1:n-1}$. When Y_n is received, we compute

$$\begin{array}{l}\mathcal{S}_n = \gamma_n \int s_n\left(\mathsf{x}_{n-1:n}\right) p_{\vartheta_{1:n}}(\mathsf{x}_{n-1:n}|\mathsf{Y}_{1:n}) d\mathsf{x}_{n-1:n} \\ + (1-\gamma_n)\sum_{m=1}^{n-1}(\prod_{i=m+1}^{n-1}(1-\gamma_i))\gamma_m \int s_m\left(\mathsf{x}_{m-1:m}\right) p_{\vartheta_{1:n}}(\mathsf{x}_{m-1:m}|\mathsf{Y}_{1:n}) d\mathsf{x}_{m-1:m},\end{array} \tag{40}$$

where the subscript $\vartheta_{1:n}$ on $p_{\vartheta_{1:n}}(\mathsf{x}_{1:T}|\mathsf{Y}_{1:n})$ indicates that the posterior density is being computed sequentially using the parameter ϑ_m at time $m \leq n$. The step sizes $\{\gamma_n\}_{n\geq 1}$ need to satisfy $\sum_n \gamma_n = \infty$ and $\sum_n \gamma_n^2 < \infty$ as in the RML case. For the M-step one uses the same maximization step (39) used in the batch version

$$\vartheta_{n+1} = \Lambda\left(\mathcal{S}_n\right). \tag{41}$$

The recursive calculation of $\mathcal{S}_n$ can be achieved by setting $V_1\left(\mathsf{x}_0\right) = 0$ and computing

$$\begin{array}{rcl} V_n\left(\mathsf{x}_n\right) & = & \int \left\{\gamma_n \ s_n\left(\mathsf{x}_{n-1}, \mathsf{x}_n\right) + \left(1-\gamma_n\right) \ V_{n-1}\left(\mathsf{x}_{n-1}\right)\right\} \\ & \times & p_{\vartheta_{1:n}}\left(\mathsf{x}_{n-1}|\, \mathsf{Y}_{1:n-1}, \mathsf{x}_n\right) d\mathsf{x}_{n-1} \end{array} \tag{42}$$

and

$$\mathcal{S}_n = \int V_n\left(\mathsf{x}_n\right) p_{\vartheta_{1:n}}(\mathsf{x}_n|\mathsf{Y}_{1:n}) d\mathsf{x}_n. \tag{43}$$

For finite state-space and linear Gaussian models, all the quantities appearing in this algorithm can be calculated exactly [20], [21], [31].

APPENDIX B
DISTRIBUTED RML DERIVATION

Let $\theta_n = \{\theta_n^{i,j}\}_{(i,j)\in\mathcal{E}}$ be the estimate of the true parameter θ_* 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: $(\dot{\mu}_{n-1}^{r,j} = \nabla_{\theta^{r,j}} \mu_{n-1}^r\big|_{\theta=\theta_n}, \mu_{n-1}^r\big|_{\theta=\theta_n}, \Sigma_{n-1}^r)$. 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. $\nabla_{\theta^{r,j}} \int x_{n-1}^r p_\theta^r(x_{n-1}^r|Y_{1:n-1}) dx_{n-1}^r\big|_{\theta=\theta_n}$. This quantity is a function of the localization parameter θ_n. Σ_{n-1}^r is the variance of the distribution

$p^r_\theta(x^r_{n-1}|Y_{1:n-1})\big|_{\theta=\theta_n}$ and is independent of the localization parameter. The log-likelihood in (27) evaluates to:

$$\log p^r_\theta(Y_n|Y_{1:n-1}) = -\frac{1}{2}\sum_{i\in\mathcal{V}}(Y^i_n - C^i_n\theta^{r,i})^T {R^i_n}^{-1}(Y^i_n - C^i_n\theta^{r,i})$$
$$-\frac{1}{2}{\mu^r_{n|n-1}}^{\mathrm{T}}(\Sigma^r_{n|n-1})^{-1}\mu^r_{n|n-1} + \frac{1}{2}(z^r_n)^{\mathrm{T}}(M^r_n)^{-1}z^r_n + const$$

where all θ 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

$$\nabla_{\theta^{r,j}}\log p^r_\theta(Y_n|Y_{1:n-1}) = -(\nabla_{\theta^{r,j}}\mu^r_{n|n-1})^{\mathrm{T}}(\Sigma^r_{n|n-1})^{-1}\mu^r_{n|n-1}$$
$$+ (\nabla_{\theta^{r,j}}z^r_n)^{\mathrm{T}}(M^r_n)^{-1}z^r_n + \sum_{i\in\mathcal{V}}(\nabla_{\theta^{r,j}}\theta^{r,i})^{\mathrm{T}}(C^i_n)^{\mathrm{T}}(R^i_n)^{-1}(Y^i_n - C^i_n\theta^{r,i}).$$

(27) requires $\nabla_{\theta^{r,j}}\log p^r_\theta(Y_n|Y_{1:n-1})$ to be evaluated at $\theta=\theta_n$. Using the equations (21)-(24) and the assumed knowledge of $(\dot\mu^{r,j}_{n-1}, \mu^r_{n-1}\big|_{\theta=\theta_n}, \Sigma^r_{n-1})$ we can evaluate the derivatives on the right-hand side of this expression:

$$\dot\mu^{r,j}_{n|n-1} = \nabla_{\theta^{r,j}}\mu^r_{n|n-1}\Big|_{\theta=\theta_n} = A_n\dot\mu^{r,j}_{n-1}, \tag{44}$$

$$\dot z^{r,j}_n = \nabla_{\theta^{r,j}}z^r_n|_{\theta=\theta_n} = (\Sigma^r_{n|n-1})^{-1}\dot\mu^{r,j}_{n|n-1} - \sum_{i\in\mathcal{V}}(C^i_n)^{\mathrm{T}}(R^i_n)^{-1}C^i_n\ \nabla_{\theta^{r,j}}\theta^{r,i}\big|_{\theta=\theta_n}, \tag{45}$$

$$\dot\mu^{r,j}_n = \nabla_{\theta^{r,j}}\mu^r_n|_{\theta=\theta_n} = (M^r_n)^{-1}\dot z^{r,j}_n. \tag{46}$$

Using property (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 $\nabla_{\theta^{r,j}}\log p^r_\theta(Y_n|Y_{1:n-1})|_{\theta=\theta_n}$ evaluates to,

$$\sum_{i\in\mathcal{V}'_{rj}}(C^i_n)^{\mathrm{T}}(R^i_n)^{-1}(Y^i_n - C^i_n\theta^{r,i}_n) = \dot m^{j,r}_{n,K} - \ddot m^{j,r}_{n,K},$$

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

$$\dot z^{r,j}_n = (\Sigma^r_{n|n-1})^{-1}\dot\mu^{r,j}_{n|n-1} - m^{j,r}_{n,K}. \tag{47}$$

To conclude, the approximations to $(\dot\mu^{r,j}_n = \nabla_{\theta^{r,j}}\mu^r_n|_{\theta=\theta_{n+1}}, \mu^r_n|_{\theta=\theta_{n+1}}, \Sigma^r_n)$ for the subsequent RML iteration, i.e. (27) at time $n=n+1$, are given by

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

while $(\mu_n^r|_{\theta=\theta_{n+1}}, \Sigma_n^r)$ are given by (21)-(24). The approximation to $\nabla_{\theta^{r,j}}\mu_n^r|_{\theta=\theta_{n+1}}$ follows from differentiating (24). $(\dot{\mu}_n^{r,j}, \mu_n^r|_{\theta=\theta_{n+1}})$ are only approximations because they are computed using the previous values of the parameters, i.e. $\theta_{1:n}$.

APPENDIX C
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,

$$Q^r(\theta_p, \theta) = \int \log p_\theta^r(x_{1:T}^r, Y_{1:T}) p_{\theta_p}^r(x_{1:T}^r|Y_{1:T}) dx_{1:T}^r,$$

$$\theta_{p+1}^{r,j} = \arg\max_{\theta^{r,j}\in\Theta} Q^r(\theta_p, (\theta^{r,j}, \{\theta^e, e \in \mathcal{E}\backslash(r,j)\})),$$

where it is assumed that node r controls edge (r, j). The quantity $p_{\theta_p}^r(x_{1:T}^r|Y_{1:T})$ 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,

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

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

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

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

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

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

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

where $(m^{j,r}_{n,K}, \dot{m}^{j,r}_{n,K}, \ddot{m}^{j,r}_{n,K})$ is defined in (18)-(20). Only $\ddot{m}^{j,r}_{n,K}$ is a function of $\theta^{r,j}$. Now to perform the M-step, we solve

$$\left(\sum_{n=1}^{T} m^{j,r}_{n,K}\right)\theta^{r,j} = \sum_{n=1}^{T}\left(\dot{m}^{j,r}_{n,K} - (m^{j,r}_{n,K})^{\mathrm{T}}\mu^{r}_{n|T} - \sum_{j'\in \mathrm{ne}(j)\setminus\{r\}} \ddot{m}^{j',j}_{n,K-1}\right).$$

Note that $\theta^{r,j}$ can be recovered by standard linear algebra and so far $\theta^{r,j}$ is solved by quantities available locally to node r and j. One can use the fact that $\sum_{j'\in \mathrm{ne}(j)\setminus\{r\}} \ddot{m}^{j',j}_{n,K-1} = \dot{m}^{j,r}_{n,K} - \dot{m}^{j,r}_{n,1}$ to so that the M-step can be performed with quantities available locally to node r only. Recall that $\sum_{n=1}^{T}\mu^{r}_{n|T} = \int\left(\sum_{n=1}^{T} x^{r}_{n}\right) p^{r}_{\theta_p}(x^{r}_{1:T}|Y_{1:T})dx^{r}_{1:T}$. This implies directly that three summary statistics are needed for node r to update $\theta^{r,j}$. These should be defined using:

$$s^{r,j}_{n,1}(x^{r}_{n}, Y_n) = (m^{j,r}_{n,K})^{\mathrm{T}}x^{r}_{n},\ s^{r,j}_{n,2}(x^{r}_{n}, Y_n) = m^{j,r}_{n,K},\ s^{r,j}_{n,3}(x^{r}_{n}, Y_n) = \dot{m}^{j,r}_{n,K} - \ddot{m}^{j,r}_{n,K} + \ddot{m}^{j,r}_{n,1},$$

where $s^{r}_{n,1}$, $s^{r}_{n,3}$ are each functions of x^{r}_{n} and Y_n via $\mu^{r}_{n|T}$ and $\dot{m}^{j,r}_{n,K} - \ddot{m}^{j,r}_{n,K} + \ddot{m}^{j,r}_{n,1}$ respectively. The summary statistics can be written in the form of (38) as follows:

$$\begin{aligned}
\mathcal{S}^{r,j^{,\theta_p}}_{T,1} &= \frac{1}{T}\int\left(\sum_{n=1}^{T}(m^{j,r}_{n,K})^{\mathrm{T}}x^{r}_{n}\right)p^{r}_{\theta_p}(x^{r}_{1:T}|Y_{1:T})dx^{r}_{1:T},\\
\mathcal{S}^{r,j^{\theta_p}}_{T,2} &= \frac{1}{T}\sum_{n=1}^{T} m^{j,r}_{n,K}, \quad \mathcal{S}^{r,j^{\theta_p}}_{T,3} = \frac{1}{T}\sum_{n=1}^{T}\left(\dot{m}^{j,r}_{n,K} - \ddot{m}^{j,r}_{n,K} + \ddot{m}^{j,r}_{n,1}\right),
\end{aligned}$$

and the M-step function becomes $\Lambda(s_1, s_2, s_3) = s_2^{-1}(s_3 - s_1)$, where s_1, s_2, s_3 correspond to each of the three summary statistics. Note that Λ is the same function for every node.

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

$$\mathcal{S}^{r,j}_{n,2} = \gamma^{r}_{n} m^{j,r}_{n,K} + (1-\gamma^{r}_{n})\,\mathcal{S}^{r,j}_{n-1,2},\ \mathcal{S}^{r,j}_{n,3} = \gamma^{r}_{n}(\dot{m}^{j,r}_{n,K} - \ddot{m}^{j,r}_{n,K} + \ddot{m}^{j,r}_{n,1}) + (1-\gamma^{r}_{n})\,\mathcal{S}^{r,j}_{n-1,3},$$

where $\{\gamma^{r}_{n}\}_{n\geq 1}$ needs to satisfy $\sum_{n\geq 1}\gamma^{r}_{n} = \infty$ and $\sum_{n\geq 1}(\gamma^{r}_{n})^2 < \infty$. For $\mathcal{S}^{r,j}_{n,1}$, we will use (42)-(43). We first set $V^{r,j}_{0}(x^{r}_{0}) = 0$ and define the recursion

$$V^{r,j}_{n}(x^{r}_{n}) = \gamma^{r}_{n}(m^{j,r}_{n,K})^{\mathrm{T}}x^{r}_{n} + (1-\gamma^{r}_{n})\int V^{r,j}_{n-1}\left(x^{r}_{n-1}\right)p^{r}_{\theta_{1:n}}\left(x^{r}_{n-1}\middle|\, Y_{1:n-1}, x^{r}_{n}\right)dx^{r}_{n-1}. \tag{48}$$

Using standard manipulations with Gaussians we can derive that $p^{r}_{\theta_{1:n}}\left(x^{r}_{n-1}\middle|\, Y_{1:n-1}, x^{r}_{n}\right)$ is itself a Gaussian density with mean and variance denoted by $\tilde{\mu}^{r}_{n}(x_n), \widetilde{\Sigma}^{r}_{n}$ respectively, where

$$\tilde{\Sigma}^{r}_{n} = \left(\Sigma^{r}_{n-1} + A^{\mathrm{T}}_{n}Q^{-1}_{n}A_n\right)^{-1},\ \widetilde{\mu}^{r}_{n}(x_n) = \tilde{\Sigma}^{r}_{n}\left(\left(\Sigma^{r}_{n-1}\right)^{-1}\mu^{r}_{n-1} + A^{\mathrm{T}}_{n}Q^{-1}_{n}x_n\right).$$

It is then evident that (48) becomes $V_n^{r,j}\left(x_n^r\right) = H_n^{r,j} x_n^r + h_n^{r,j}$, with:

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

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

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

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

ACKNOWLEDGEMENT

N. Kantas was supported by the Engineering and Physical Sciences Research Council programme grant on Control For Energy and Sustainability (EP/G066477/1). S.S. Singh's research is partly funded by the Engineering and Physical Sciences Research Council under the First Grant Scheme (EP/G037590/1).

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 Processing Magazine*, vol. 22, no. 4, pp. 54–69, July 2005.

[2] K. Plarre and P. Kumar, "Tracking objects with networked scattered directional sensors," *EURASIP Journal on Advances in Signal Processing*, 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 Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM 2005*, vol. 1, 13–17 March 2005, 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 Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 809–819, April 2005.

[5] R. L. Moses, D. Krishnamurthy, and R. Patterson, "A self-localization method for wireless sensor networks," *Eurasip Journal on Applied Signal Processing, Special Issue on Sensor Networks*, 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 Processing*, pp. 1144–1154, 2009.

[7] V. Cevher and J. H. McClellan, "Acoustic node calibration using moving sources," *IEEE Transactions on Aerospace and Electronic Systems*, 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 Information Processing in Sensor Networks, Chicago, IL*, 2011.

[9] S. Funiak, C. Guestrin, M. Paskin, and R. Sukthankar, "Distributed localization of networked cameras," in *Proc. Fifth International Conference on Information Processing in Sensor Networks 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. Fifth International Conference on Information Processing in Sensor Networks IPSN 2006*, 2006, pp. 27–33.

[11] Y. Baryshnikov and J. Tan, "Localization for anchoritic sensor networks," in *Proc. 3rd IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS '07)*, Santa Fe, New Mexico, USA, 18–20 June 2007.

[12] N. Kantas, S. S. Singh, and A. Doucet, "Distributed online self-localization and tracking in sensor networks," in *Proc. 5th International Symposium on Image and Signal Processing and Analysis ISPA 2007*, 27–29 Sept. 2007, pp. 498–503.

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

[14] N. Kantas, "Stochastic decision making for general state space models," Ph.D. dissertation, University of Cambridge, February 2009.

[15] O. Cappé, E. Moulines, and T. Rydén, *Inference in hidden Markov models*. Springer, 2005.

[16] R. Elliott, L. Aggoun, and J. Moore, *Hidden Markov models: estimation and control*. Springer-Verlag, 1995.

[17] U. Holst and G. Lindgren, "Recursive estimation in mixture models with markov regime," *IEEE Transactions on Information Theory*, vol. 37, no. 6, pp. 1683–1690, Nov. 1991.

[18] F. LeGland and L. Mevel, "Recursive estimation in hidden markov models," in *Proc. 36th IEEE Conference on Decision and Control*, vol. 4, 10–12 Dec. 1997, pp. 3468–3473.

[19] I. B. Collings and T. Ryden, "A new maximum likelihood gradient algorithm for on-line hidden markov model identification," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 4, 12–15 May 1998, pp. 2261–2264.

[20] J. J. Ford, "Adaptive hidden markov model estimation and applications," Ph.D. dissertation, Dept. of Systems Engineering, Australian National University, 1998.

[21] R. Elliott, J. Ford, and J. Moore, "On-line consistent estimation of hidden markov models," Department of Systems Engineering, Australian National University, Tech. Rep., 2000.

[22] O. Cappe and E. Moulines, "Online em algorithm for latent data models," *Journal of the royal statistical society. Series B (Methodological)*, vol. 71, p. 593, 2009.

[23] R. D. Nowak, "Distributed em algorithms for density estimation and clustering in sensor networks," *IEEE Transactions on Signal Processing, Special Issue on Signal Processing in Networking*, vol. 51, no. 8, pp. 2245–2253, Aug. 2003.

[24] M.-A. Sato and S. Ishii, "On-line EM Algorithm for the Normalized Gaussian Network," *Neural Comp.*, vol. 12, no. 2, pp. 407–432, 2000.

[25] D. Blatt and A. Hero, "Distributed maximum likelihood estimation for sensor networks," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, 2004, pp. 929–932.

[26] N. N. Okello and S. Challa, "Joint sensor registration and track-to-track fusion for distributed trackers," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 3, pp. 808–823, July 2004.

[27] J. Vermaak, S. Maskell, and M. Briers, "Online sensor registration," in *Proc. IEEE Aerospace Conference*, 5–12 March 2005, pp. 2117–2125.

[28] L. Xiao, S. Boyd, and S. Lall, "A scheme for robust distributed sensor fusion based on average consensus," in *Proc. Fourth International Symposium on Information Processing in Sensor Networks IPSN 2005*, 15 April 2005, pp. 63–70.

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

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

[31] O. Cappe, "Online em algorithm for hidden markov models," *Journal Computational Graphical Statistics*, vol. to appear, 2011.

[32] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *In Proc. 44th Annual IEEE Symposium on Foundations of Computer Science, 2003.*, 2003, pp. 482–491.

[33] D. Bertsekas and J. Tsitsiklis, *Parallel and distributed computation*. Prentice Hall Inc.m, Old Tappan, NJ (USA), 1989.

[34] D. M. Titterington, "Recursive parameter estimation using incomplete data," *Journal of the Royal Statistical Society of London, Series B (Methodological)*, vol. 46, no. 2, pp. 257–267, 1984.

[35] D. M. Titterington and J.-M. Jiang, "Recursive estimation procedures for missing-data problems," *Biometrika*, vol. 70, no. 3, pp. 613–624, Dec 1983.

[36] A. P. Dempster, N. M. Laird, and D. B. Rudin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society of London, Series B (Methodological)*, vol. 39, pp. 1–38, 1977.

[37] R. J. Elliott, J. J. Ford, and J. B. Moore, "On-line almost-sure parameter estimation for partially observed discrete-time linear systems with known noise characteristics," *International Journal of Adaptive Control and Signal Processing*, vol. 16, no. 6, pp. 435–453, 2002.