Distributed Online Self-Localisation and Tracking in Sensor Networks

Nikolas Kantas Sumeetpal S. Singh[*]
Signal Processing Group,
Department of Engineering
University of Cambridge, UK
{nk234,sss40}@cam.ac.uk

Arnaud Doucet
Department of Computer Science and
Department of Statistics
University of British Columbia, CA
arnaud@cs.ubc.ca

Abstract

Recursive Maximum Likelihood (RML) and Expectation Maximisation (EM) are a popular methodologies for estimating unknown static parameters in state-space models. We describe how a completely decentralized version of RML and EM can be implemented in dynamic graphical models through the propagation of suitable messages that are exchanged between neighbouring nodes of the graph. The resulting algorithm can be interpreted as an extension of the celebrated Belief Propagation algorithm to compute likelihood gradients. This algorithm is applied to solve the sensor localisation problem for sensor networks. An exact implementation is given for dynamic linear Gaussian models without loop. For non-linear scenarios, a distributed extended Kalman filter is used to implement RML.

1 Introduction

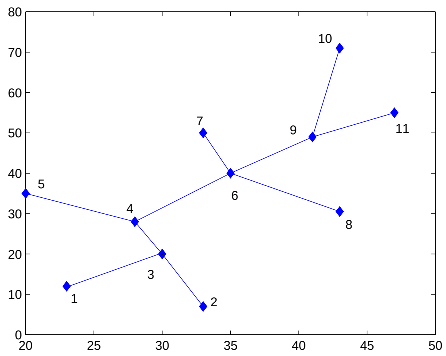


Figure 1. Sensor Network used for target tracking

Figure 1 depicts a sensor network that is deployed to perform target tracking. The sensor network is comprised of sensor-trackers where each node in the network has the processing ability to perform computations needed for target tracking.[1] The lines joining the nodes indicates communication links and defines the neighborhood structure of the network. A sensor can only communicate indirectly with non-neighboring nodes. A target that is traversing the field of view of the network will be simultaneously observed by more than one sensors, depending of course on their field-of-view. In a centralised architecture all the sensors transmit their measurements to a central fusion node which then performs all the computation required [4]. Our aim is to perform *collaborative tracking* but without the need for a central fusion node. In such networks nodes collaborate by exchanging appropriate messages between neighbouring nodes to achieve the same net effect as they would by communicating with a central fusion node.

For generality, we assume that each node maintains a local coordinate system (or frame of reference) and regards itself as the origin (or centre) its coordinate system. Distributed collaborative tracking can be achieved if each node is able to accurately determine the position of its neighbouring nodes in its local frame of reference. (More details in Section 2.) This is essentially an instance of the *self-localisation* problem [2]. In the interest of generality and scalability of the proposed solution, in this paper we simultaneously solve the localisation problem as well. We will do so without the need of a Global Positioning System (GPS) or direct measurements of the distance between neighbouring nodes. The latter is usually estimated from the Received Signal Strength (RSS) when each node is equipped with a wireless transceiver. The method we propose is significantly different. Essentially, the very task that the network is deployed for, which is collaborative tracking, will be exploited to achieve self-localization in a completely decentralized manner. Initially as nodes are not localized they 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 away that allows them to self-localize. Even though the target's true trajectory is not known to the sensors, localization can be

[*]The research of the second author was funded by Boeing. Part of the work in this paper was undertaken while S.S. Singh was a visiting researcher at National ICT Australia Ltd., Victorial Labs, Australia, in January 2007. Financial support from NICTA during this period is gratefully acknowledged.

[1]It should be noted that we are not dealing with simple sensors with limited processing power.

achieved in this manner because the same target is being simultaneously measured by the sensors.

The method we propose in this paper appears to have been independently developed by [7, 16]. However, our work differs from these in application, the inference scheme and the computational tools employed. Both [7, 16] formulate the localisation as a posterior inference problem and use Gaussian approximation schemes as the computational tool. [7] studies a computer vision based tracking problem. We propose to use likelihood inference techniques, namely Recursive Maximum Likelihood (RML) and Expectation-Maximization (EM). These techniques have not been developed for the self-localisation problem and our results in this paper address this shortfall. The RML and EM based algorithms we develop admit an online implementation which is important for the application studied. Most tracking problems are essentially non-linear non-Gaussian filtering problems and Sequential Monte Carlo (SMC) methods, also known as Particle Filters, provide very good approximations to the filtering densities under weak assumptions [6]. While it is possible to develop SMC versions of our algorithms, because of execution speed, we opt instead for linearisation (as in the Extended Kalman filter) when dealing with a non-linear system.

The decentralized solution to the self-localization and collaborative tracking problem necessitates the use of *belief propagation*, which a message passing algorithm widely used in the computer science literature to perform posterior inference on graphs [13]. However belief propagation implements posterior inference only while we also require a fully decentralized algorithm for calculating the gradient of the log-likelihood function. We propose a message passing scheme similar to belief propagation for doing so which is, to the best of our knowledge, novel.

There is a vast literature on the self-localisation problem. Furthermore, the topic has been independently pursued by researchers working in different application areas, most notably wireless communications [12, 14, 8] and sensor networks for environmental monitoring [10]. Although all these works tend to be targeted for the application at hand and differ in implementation specifics, which we do not wish to list here, they may however be broadly characterise as follows. There are many works the rely on direct measurements between neighbouring nodes [12, 14, 8]. 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. There are non-statistical based approaches of this idea, like least squares [14], and statistical ones based on probabilistic inference [8], likelihood [12]. There are also methods that utilise *beacon* nodes which have either been manually placed at precise locations or are equipped with a GPS. The un-localised nodes will use the signal broadcast by these beacon nodes to self-localize [12].

There is the related problem of *sensor registration* which aims to compensate for systematic biases in the sensors has been studied by the target tracking community [11, 17]. The algorithms devised therein are for are all centralised. There is also the related problem of average consensus [18]. The value of a global static parameter is measured at each node via a linear Gaussian observation model and the aim to obtain a maximum likelihood estimate in distributed fashion. This is a distributed averaging problem and not a distributed localization and tracking task. A second paper that is based on the principle shared by our approach and [7, 16] is that of [3]. The authours 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 centralised in nature.

2 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. In this paper we assume that the sensor network has no loops. Nodes $i, j \in \mathcal{V}$ are connected provided the edge $(i, j) \in \mathcal{E}$ exists. All nodes are time synchronised distributed trackers and observe the same physical target at discrete time intervals $n = 1, 2, \ldots$. The non-linear non-Gaussian setting can be expressed with the following Hidden Markov Model (HMM)[2],

$$X_n^r | x_{n-1}^r \sim f_n(.|x_{n-1}^r), Y_n^r | x_n^r \sim g_n^r(.|x_n^r). \qquad (1)$$

where $X_n^r \in \mathbb{R}^{d_x}$ is the hidden state and $Y_n^r \in \mathbb{R}^{d_y^r}$ is the measurement made by node r at time n. A common state-space $\mathbb{R}^{d_x}$ and transition model $f_n(.|x_{n-1}^r)$ is adopted for all nodes $r \in \mathcal{V}$ since they track the same physical target. The vector X_n^r at node r is defined relative to the local coordinate system at node r which regards itself as the origin. The hidden state in standard target tracking examples is defined to comprise of the position and velocity of the target,

$$X_n^r = [X_n^r(1), X_n^r(2), X_n^r(3), X_n^r(4)]^{\mathrm{T}} \in \mathbb{R}^4,$$

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. The target's transition model $f_n(.|.)$ is assumed time varying for generality. In specific examples in Section 5 adopt the same definition for the hidden state.

The measurement Y_n^r made by node r is also defined relative to the local coordinate system at node r. The observation model is time varying and is different for each node. Also, the length 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.

[2]Subscripts on a variable always indicate time while a superscript will indicate the node the quantity belongs to or the edge in the case of $\theta^{i,j}$.

Assuming all nodes act independently the *filtering density*, which summarises all relevant information of hidden state given by all the observations that have been received thus far, is propagated at each node locally using the Bayes recursion,

$$\text{prediction:} p^r(x_n^r|Y_{1:n-1}^r) = \int f_n(x_n^r|x_{n-1}^r) p^r(x_{n-1}^r|Y_{1:n-1}^r) dx_{n-1}^r \quad (2)$$

$$\text{update:} p^r(x_n^r|Y_{1:n}^r) \propto g_n^r(Y_n^r|x_n^r) p^r(x_n^r|Y_{1:n-1}^r) \quad (3)$$

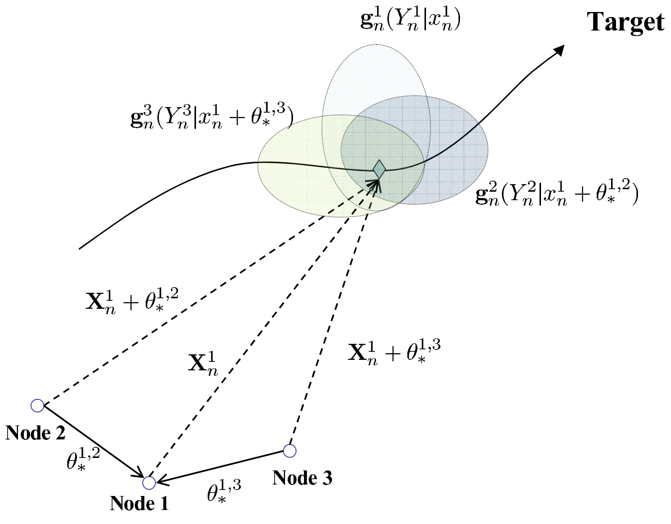
A linear Gaussian system is a special case where the transition and observation densities are Gaussian. The target being tracked follows the linear Gaussian model

$$X_n^r = A_n X_{n-1}^r + b_n + V_n^r, \quad n \geq 1, \quad (4)$$

where V_n^r is zero mean Gaussian additive noise with variance Q_n, b_n are deterministic inputs. As before a common state-space and a common time varying target model $\{(A_n, b_n, Q_n)\}_{n\geq 1}$ applies to all nodes $r \in \mathcal{V}$. At each node r, the following linear Gaussian observation model applies,

$$Y_n^r = C_n^r X_n^r + d_n^r + W_n^r, \quad n \geq 1, \quad (5)$$

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 for the non-linear setting. It is in this setting as well that the need for sequences $\{b_n\}_{n\geq 1}$ and $\{d_n^r\}_{n\geq 1}$ arises. In the development below, to simplify the presentation, we ignore the sequences $\{b_n\}_{n\geq 1}$ and $\{d_n^r\}_{n\geq 1}$ for all $r \in \mathcal{V}$.



$\mathbf{g}_n^1(Y_n^1|x_n^1)$
Target
$\mathbf{g}_n^3(Y_n^3|x_n^1+\theta_*^{1,3})$
$\mathbf{g}_n^2(Y_n^2|x_n^1+\theta_*^{1,2})$
$\mathbf{X}_n^1+\theta_*^{1,2}$
$\mathbf{X}_n^1$
$\mathbf{X}_n^1+\theta_*^{1,3}$
Node 2
$\theta_*^{1,2}$
Node 1
$\theta_*^{1,3}$
Node 3

Figure 2. Three nodes jointly observing a target

Figure 2 illustrates a three node setting where a jointly observed target is being tracked by three sensor nodes defined on the same state space $\mathbb{R}^{d_x}$ with $d_x = 4$. (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 2),

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

The *localisation* parameters $\{\theta_*^{i,j}\}_{(i,j)\in\mathcal{E}}$ are static if 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

$$\theta_*^{i,j} = \theta_*^{i,j_1} + \theta_*^{j_1,j_2} + \theta_*^{j_2,j_3} + \ldots + \theta_*^{j_{m-1},j_m} + \theta_*^{j_m,j}. \quad (6)$$

This relationship is key to deriving 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 as, only the first and third component of $\theta_*^{i,j}$ is relevant while $\theta_*^{i,j}(1) = \theta_*^{i,j}(2)$ and $\theta_*^{i,j}(3) = \theta_*^{i,j}(4)$. Let

$$\theta_* \equiv \{\theta_*^{i,j}\}_{(i,j)\in\mathcal{E}}, \quad \theta_*^{i,i} \equiv 0, \quad (7)$$

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

Returning to Figure 2 ,the filtering update step at node 1 can be refined with the observations made by nodes 2 and 3 provided the localisation parameters $\theta_*^{1,2}$ and $\theta_*^{1,3}$ are known locally to node 1 since $p_{\theta_*}^1(Y_n|x_n^1)$ is

$$g_n^1(Y_n^1|x_n^1) g_n^2(Y_n^2|x_n^1 + \theta_*^{1,2}) g_n^3(Y_n^3|x_n^1 + \theta_*^{1,3}),$$

where $Y_n \equiv \{Y_n^v\}_{v\in\mathcal{V}}$.

Figure 2 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. Additionally, it appears that node 1 would also require knowledge of the functions $g_n^3(Y_n^3|.)$ and $g_n^3(Y_n^3|.)$. In general, in the *collaborative* filtering problem, each node r propagates

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

where $p_{\theta_*}^r(x_n^r|Y_{1:n-1}^r)$ is given in (2), while the likelihood term is

$$p_{\theta_*}^r(Y_n|x_n^r) = \prod_{v\in\mathcal{V}} g_n^v(Y_n^v|x_n^r + \theta_*^{r,v}). \quad (8)$$

where $\theta_*^{v,v}$ for all $v \in \mathcal{V}$ is defined to be the zero vector by default. The prediction step is unchanged (can be implemented locally at each node without exchange of information) but the update step now incorporates all the measurements of the network. The term collaborative filtering is used since each sensor benefits from the observation made by all the other sensors. As is shown in Section 3, it is possible to implement collaborative filtering in a truly distributed manner, i.e., each node executes a message passing algorithm (with communication limited only to neighbouring nodes) that is scalable with the size of the network. However collaborative filtering hinges on knowledge of the localisation parameters $\{\theta_*^{i,j}\}_{(i,j)\in\mathcal{E}}$ which are unknown *apriori*. We propose a estimation algorithms based on RML and EM that simultaneously learn the localisation parameters. RML is an online algorithm and refines the parameter estimates as new data arrives while EM is a batch algorithm that runs once a batch of observations have been acquired. These proposed algorithms are to the best of our knowledge novel.

3 Distributed Collaborative Filtering and Smoothing

For a linear Gaussian system, the collaborative filter π_n^v at node v is a Gaussian distribution with mean vector μ_n^v and covariance matrix Σ_n^v. Consider a fixed localisation parameter $\theta = \{\theta^{i,j}\}_{(i,j)\in\mathcal{E}}$. Each node v of the network will execute the following Kalman filter whose derivation is standard,

$$\mu_{n|n-1}^v = A_n\mu_{n-1}^v, \Sigma_{n|n-1}^v = A_n\Sigma_{n-1}^v A_n^{\mathrm{T}} + Q_n, \quad (9)$$

$$M_n^v = (\Sigma_{n|n-1}^v)^{-1} + \sum_{i\in\mathcal{V}} (C_n^i)^{\mathrm{T}}(R_n^i)^{-1}C_n^i, \quad (10)$$

$$z_n^v = (\Sigma_{n|n-1}^v)^{-1}\mu_{n|n-1}^v + \sum_{i\in\mathcal{V}} (C_n^i)^{\mathrm{T}}(R_n^i)^{-1}(Y_n^i - C_n^i\theta^{v,i}), \quad (11)$$

$$\Sigma_n^v = (M_n^v)^{-1}, \quad \mu_n^v = \Sigma_n^v z_n^v. \quad (12)$$

The non-local steps are the recursions for M_n^v and z_n^v. The recursion for z_n^v requires $\sum_{i\in\mathcal{V}}(C_n^i)^{\mathrm{T}}(R_n^i)^{-1}Y_n^i$ and $\sum_{i\in\mathcal{V}}(C_n^i)^{\mathrm{T}}(R_n^i)^{-1}C_n^i\theta^{v,i}$ to be available locally. Similarly for M_n^v. Owing to property (6), the following message passing scheme achieves the desired distributed implementation.

$$m_n^{i,j} = (C_n^i)^{\mathrm{T}}(R_n^i)^{-1}C_n^i + \sum_{p\in\mathrm{ne}(i)\setminus\{j\}} m_n^{p,i}, \quad (13)$$

$$\dot{m}_n^{i,j} = (C_n^i)^{\mathrm{T}}(R_n^i)^{-1}Y_n^i + \sum_{p\in\mathrm{ne}(i)\setminus\{j\}} \dot{m}_n^{p,i}, \quad (14)$$

$$\ddot{m}_n^{i,j} = m_n^{i,j}\theta^{j,i} + \sum_{p\in\mathrm{ne}(i)\setminus\{j\}} \ddot{m}_n^{p,i}, \quad (15)$$

where ne(i) denote the neighbours of node i. A message from node i to j can be sent once node i has received message from all its neighbours except node i. Thus the leaf nodes (nodes with only one neighbour) of the network will originate the messages. Note that these messages are matrix and vector valued quantities and they require a fixed amount memory for storage regardless of the number of nodes in the network. Also, the same rule for generating and combining messages are implemented at each node. Once a node has received a message from all its neighbours it can perform the recursions for M_n^v and z_n^v since

$$M_n^v = (\Sigma_{n|n-1}^v)^{-1} + (C_n^v)^{\mathrm{T}}(R_n^v)^{-1}C_n^v + \sum_{i\in\mathrm{ne}(v)} m_n^{i,v} \quad (16)$$

$$z_n^v = (\Sigma_{n|n-1}^v)^{-1}\mu_{n|n-1}^v + (C_n^v)^{\mathrm{T}}(R_n^v)^{-1}Y_n^v + \sum_{i\in\mathrm{ne}(v)} \dot{m}_n^{i,v} - \ddot{m}_n^{i,v} \quad (17)$$

For collaborative smoothing, once a batch of T observations have been obtained, each node r aims to implement

$$\pi_{n|T}^r(x_n^r) \propto \int p_T^r(x_{1:T}^r, Y_{1:T})dx_{1:T\setminus\{n\}}^r$$

where $dx_{1:T\setminus\{n\}}^r$ means integration w.r.to all variables except x_n^r. Naturally $\pi_{T|T}^r$ is just the filtering density at time T. The standard Kalman smoother is implemented with a forward pass (9)-(12) , which computes the usual filtering densities, and the followed by a backward pass which is summarised by the following equations [15]:

$$J_{n-1}^r = \Sigma_{n-1}^r A_n^{\mathrm{T}}(\Sigma_{n|n-1}^r)^{-1} \quad (18)$$

$$\mu_{n-1|T}^r = \mu_{n-1}^r + J_{n-1}^r(\mu_{n|T}^r - A_n\mu_{n-1}^r) \quad (19)$$

$$\Sigma_{n-1|T}^r = \Sigma_{n-1}^r + J_{n-1}^r(\Sigma_{n|T}^r - \Sigma_{n|n-1}^r)J_{n-1}^{r\,\mathrm{T}} \quad (20)$$

The backward pass is performed commencing with $n = T$ until $n = 2$ and the smoothed density is $\pi_{n|T}^r = \mathcal{N}(\mu_{n|T}^r, \Sigma_{n|T}^r)$. It is an entirely local procedure with no exchange of information between neighbouring nodes necessary.

4 Distributed Collaborative Lozalisation

Our sensor localisation problem is a static parameter estimation problem as discussed in Section 2. To solve the localization problem, we propose to use likelihood inference techniques, namely Recursive Maximum Likelihood and Expectation-Maximization. RML is a stochastic gradient algorithm that maximises the *average log likelihood* and is given by the following recursion [9]:

$$\theta_{n+1} = \theta_n + \gamma_{n+1}\left[\nabla \log p_\theta(Y_n|Y_{1:n-1})\right]|_{\theta=\theta_n}, \quad (21)$$

where the gradient (denoted by ∇) is taken w.r.to the parameter θ, $\{\gamma_n\}$ is a sequence of small positive real numbers (e.g. $\gamma_n = n^{-1}$), called the step-size sequence, that satisfies $\sum_n \gamma_n = \infty$ and $\sum_n {\gamma_n}^2 < \infty$. θ_n is the estimate of the true parameter θ_* given the available data $Y_{1:n-1}$. Upon receiving Y_n, an increment is added to θ_n in the direction of ascent of the conditional likelihood $\log p_\theta(Y_n|Y_{1:n-1})$. In practise, a small but fixed step-size is used is used as it ensures continual adaptation.

For a given sequence of T observations, the EM algorithm for learning θ_* is a two step procedure. The first step, the expectation or E-step, computes

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

The second step is the maximisation or M-step that updates the parameter θ_n,

$$\theta_{n+1} = \arg\max_{\theta\in\Theta} Q(\theta_n, \theta)$$

Upon the completion of an E and M step, the likelihood surface is ascended or $p_{\theta_{n+1}}(Y_{1:T}) \geq p_{\theta_n}(Y_{1:T})$ [5]. For linear Gaussian state-space models this procedure can be implemented exactly while for the general non-linear non-Gaussian case SMC approximations are used. The RML and EM algorithm described thus far is centralized and we now detail its distributed implementation.

4.1 Distributed RML

Figure 2 shows a particular node r in a network which has control over the edge (r, j). Because every edge is assigned to one node, all edge-controlling nodes will implement a RML algorithm to learn the θ-parameter for its edge. Let $\theta_n = \{\theta_n^{i,j}\}_{(i,j)\in\mathcal{E}}$ be the estimate of the true parameter θ_* given the available data $Y_{1:n-1}$. Recall that Y_m is defined to be $\{Y_m^v\}_{v\in\mathcal{V}}$ so $Y_{1:n-1}$ is the data collected by all the nodes from time 1 to $n-1$. At a given node r that controls edge (r, j) the following RML algorithm is implemented,

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

where the gradient is computed w.r.to $\theta^{r,j}$,

$$p_\theta^r(x_n^r|Y_{1:n-1}) = \int f_n(x_n^r|x_{n-1}^r) p_\theta^r(x_{n-1}^r|Y_{1:n-1}) dx_{n-1}^r$$

is the local collaborative *predicted* density at node r, which is a function of $\theta = \{\theta^{i,j}\}_{(i,j)\in\mathcal{E}}$, and likelihood term is (see (8)),

$$p_\theta^r(Y_n|x_n^r) = \prod_{v\in\mathcal{V}} g_n^v(Y_n^v|x_n^r + \theta^{r,v}). \tag{22}$$

Note that node r updates $\theta_n^{r,j}$ in a direction of increase of the log-likelihood of $Y_n = \{Y_n^v\}_{v\in\mathcal{V}}$ given all the measurements received in the network from time 1 to $n-1$. Thus the RML procedure is truly collaborative. 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 that control an edge parameter will implement such a local gradient algorithm.

For the linear Gaussian case, we have

$$\begin{aligned}
&\log p_\theta^r(Y_n|Y_{0:n-1}) \\
&= -\frac{1}{2}\sum_{i\in\mathcal{V}}(Y_n^i - C_n^i\theta^{r,i})^T {R_n^i}^{-1}(Y_n^i - C_n^i\theta^{r,i}) \\
&- \frac{1}{2}{\mu_{n|n-1}^r}^{\mathrm{T}}(\Sigma_{n|n-1}^r)^{-1}\mu_{n|n-1}^r \\
&+ \frac{1}{2}(z_n^r)^{\mathrm{T}}(M_n^r)^{-1}z_n^r + const
\end{aligned}$$

where all θ-independent terms have been lumped together in the term 'const'. Differentiating this expression w.r.to $\theta^{r,j}$ yields

$$\begin{aligned}
&\nabla_{\theta^{r,j}} \log p_\theta^r(Y_n|Y_{0:n-1}) \\
&= -(\nabla_{\theta^{r,j}}\mu_{n|n-1}^r)^{\mathrm{T}}(\Sigma_{n|n-1}^r)^{-1}\mu_{n|n-1}^r \\
&+ (\nabla_{\theta^{r,j}}z_n^r)^{\mathrm{T}}(M_n^r)^{-1}z_n^r \\
&+ \sum_{i\in\mathcal{V}}(\nabla_{\theta^{r,j}}\theta^{r,i})^{\mathrm{T}}(C_n^i)^{\mathrm{T}}(R_n^i)^{-1}(Y_n^i - C_n^i\theta^{r,i}).
\end{aligned}$$

Using the recursion of (9)-(12) we can to propagate terms $\nabla_{\theta^{r,j}}\mu_{n|n-1}^r$, $\nabla_{\theta^{r,j}}z_n^r$ and $\nabla_{\theta^{r,j}}\mu_n^v$ locally each node r as follows:

$$\nabla_{\theta^{r,j}}\mu_{n|n-1}^r = A_n\nabla_{\theta^{r,j}}\mu_{n-1}^r, \tag{23}$$

$$\nabla_{\theta^{r,j}}z_n^r = (\Sigma_{n|n-1}^r)^{-1}\nabla_{\theta^{r,j}}\mu_{n|n-1}^r - \sum_{i\in\mathcal{V}}(C_n^i)^{\mathrm{T}}(R_n^i)^{-1}C_n^i\nabla_{\theta^{r,j}}\theta^{r,i}, \tag{24}$$

$$\nabla_{\theta^{r,j}}\mu_n^r = (M_n^r)^{-1}\nabla_{\theta^{r,j}}z_n^r. \tag{25}$$

Using property (6) 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, for the last sum in the expression for $\nabla_{\theta^{r,j}} \log p_\theta^r(Y_n|Y_{0:n-1})$ evaluates to,

$$\sum_{i\in\mathcal{V}'_{rj}}(C_n^i)^{\mathrm{T}}(R_n^i)^{-1}(Y_n^i - C_n^i\theta^{r,i}) = \dot{m}_n^{j,r} - \ddot{m}_n^{j,r}.$$

Similarly, we can write the sum in the expression for $\nabla_{\theta^{r,j}}z_n^r$ as

$$\sum_{i\in\mathcal{V}}(C_n^i)^{\mathrm{T}}(R_n^i)^{-1}C_n^i\nabla_{\theta^{r,j}}\theta^{r,i} = \sum_{i\in\mathcal{V}'_{rj}}(C_n^i)^{\mathrm{T}}(R_n^i)^{-1}C_n^i = m_n^{j,r}.$$

This derivation is now summarised in the following algorithm.

Distributed Filtering and Parameter Estimation With RML

At time n, upon receiving $Y_n = \{Y_n^v\}_{v\in\mathcal{V}}$ do:

Prediction for all nodes: At each node $v \in \mathcal{V}$ compute locally the filtering prediction step of equation (2). For the linear Gaussian problem the prediction is as in (9). At the same time propagate the gradient $\nabla_{\theta^{r,j}}\mu_{n|n-1}^r$ as in (23).

Propagate messages: After all observations in Y_n are received from all nodes propagate all messages $m_n^{i,j}$, $\dot{m}_n^{i,j}$, $\ddot{m}_n^{i,j}$. These are given by (13)-(15). Note that now $\ddot{m}_n^{i,j}$ is a function of θ_n.

Filtering Update: At each node v perform the filtering update step of (3) by computing (12), where M_n^v and z_n^v are given in (16)-(17). In parallel propagate the gradients as in equations (24)-(25).

Gradient update of each $\theta_{n+1}^{r,j}$: At each node r controlling edge (r,j) set

$$\theta_{n+1}^{r,j} = \theta_n^{r,j} + \gamma_{n+1}(-\nabla_{\theta^{r,j}} {\mu_{n|n-1}^r}^{\mathrm{T}} (\Sigma_{n|n-1}^r)^{-1} \mu_{n|n-1}^r + \nabla_{\theta^{r,j}} {z_n^r}^{\mathrm{T}} (M_n^r)^{-1} z_n^r + \dot{m}_n^{j,r} - \ddot{m}_n^{j,r}).$$

4.1.1 Non-linear Model

The (Extended Kalman Filter) EKF implementation for distributed RML is a based on local linearisation. Let the distributed tracking system be given by the following model:

$$X_n^r = \phi_n(X_{n-1}^r) + V_n^r, Y_n^r = \psi_n^r(X_n^r) + W_n^r. \quad (26)$$

Let iteration n, prior to receiving $\{Y_n^v\}_{v\in\mathcal{V}}$, let $\theta_n = \{\theta_n^{i,j}\}_{(i,j)\in\mathcal{E}}$ be the estimate of the true localisation parameter θ_* given the available data $Y_{1:n-1}$. Each node upon completion of the local prediction steps linearises its state and observation model about the predicted mean, i.e., a given node r will implement ,

$$\begin{aligned} X_n^r &= \phi_n(\mu_{n-1}^r) \\ &\quad + \nabla\phi_n(\mu_{n-1}^r)(X_{n-1}^r - \mu_{n-1}^r) + V_n^r, \quad (27) \\ Y_n^r &= \psi_n^r(\mu_{n|n-1}^r) \\ &\quad + \nabla\psi_n^r(\mu_{n|n-1}^r)(X_n^r - \mu_{n|n-1}^r) + W_n^r. \quad (28) \end{aligned}$$

where for a mapping $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$, $\nabla f \equiv [\nabla f_1, \ldots, \nabla f_d]^{\mathrm{T}}$. Note that after linearisation extra additive terms appear as seen in the general setting described by equations (4)-(5). As far as the filtering prediction step is concerned the only alteration in the algorithm of Section 4.1 will appear in equation (9), where in the rhs of the equation for $\mu_{n|n-1}^r$, $\phi_n(\mu_{n-1}^r) - \nabla\phi_n(\mu_{n-1}^r)\mu_{n-1}^r$ should be added. For the rest of the steps concerning the filtering update and the gradient update of each parameter the setting of equation (5) is preserved if Y_n^r is substituted by $Y_n^r - \psi_n^r(\mu_{n|n-1}^r) + \nabla\psi_n^r(\mu_{n|n-1}^r)\mu_{n|n-1}^r$. Otherwise the algorithm remains the same.

4.2 Distributed EM

For the 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,

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

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

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

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

where $p_\theta^r(Y_n|x_n^r)$ was defined in (8). Note that $p_{\theta_n}^r(x_{1:T}^r, Y_{1:T})$ (and hence $p_{\theta_n}^r(x_{1:T}^r|Y_{1:T})$) is a function of $\theta_n = \{\theta_n^{i,i'}\}_{(i,i')\in\mathcal{E}}$ and not just $\theta_n^{r,j}$. Also, the θ-dependance 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

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

where c_n^v is a constant independent of θ. Using the fact that $x^{\mathrm{T}}Ax = trace(Axx^{\mathrm{T}})$ and $E(x^{\mathrm{T}}Ax) = trace(AE(xx^{\mathrm{T}}))$, taking the expectation w.r.to $p_{\theta_n}^r(x_n^r|Y_{1:T})$ gives

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

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 $p_{\theta_n}^r(x_n^r|Y_{1:T})$. Taking the gradient w.r.to $\theta^{r,j}$ we get and following the steps in the derivation of the distributed RML we obtain

$$\begin{aligned} &\nabla_{\theta^{r,j}} \int \log p_\theta^r(Y_n|x_n^r) p_{\theta_n}^r(x_n^r|Y_{1:T}) \\ &= \dot{m}_n^{j,r} - \ddot{m}_n^{j,r} - (m_n^{j,r})^{\mathrm{T}} \mu_{n|T}^r \end{aligned}$$

where $(m_n^{j,r}, \dot{m}_n^{j,r}, \ddot{m}_n^{j,r})$ is defined in (13)-(15). Only $\ddot{m}_n^{j,r}$ is a function of $\theta^{r,j}$. Now to perform the M-step, we solve

$$\begin{aligned} &\left(\sum_{n=1}^T m_n^{j,r}\right)\theta^{r,j} \\ &= \sum_{n=1}^T \left(\dot{m}_n^{j,r} - (m_n^{j,r})^{\mathrm{T}} \mu_{n|T}^r - \sum_{p\in\text{ne}(j)\backslash\{r\}} \ddot{m}_n^{p,j}\right) \end{aligned}$$

and $\theta^{r,j}$ can recovered by standard linear algebra. Note that $\theta^{r,j}$ is solved by quantities available locally to node r and j only.

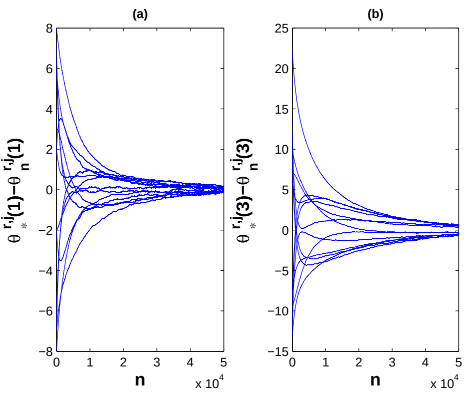


Figure 3. Error at each RML iteration between the true parameter and current update of the localisation parameter for all edges of the sensor net. (a) and (b) show the errors in the x- and y- coordinates respectively.

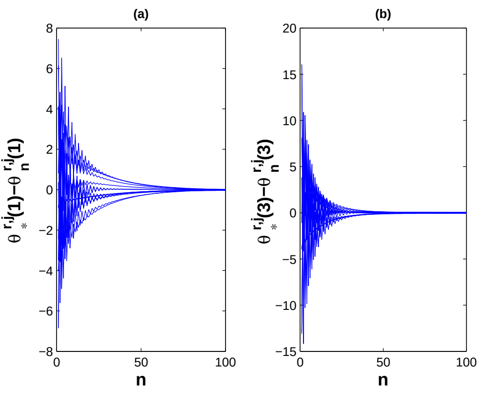


Figure 4. Error at each EM iteration between the true parameter and current update of the localisation parameter for all edges of the sensor net. (a) and (b) show the errors in the x- and y- coordinates respectively.

5 Numerical Examples

We would like to illustrate the performance of the distributed RML algorithm for the Linear Gaussian case in a numerical example. The sensor network in Figure 1 has a tree structure and we apply distributed RML as in the previous section to the self localisation problem. We choose nodes $\{3, 4, 6, 9\}$ as root nodes and update at each iteration their adjacent edges. For practical implementation reasons we choose to use a constant step size $\gamma_k = 10^{-3}$. For stochastic approximation in general, decreasing step-sizes are essential conditions of convergence. If fixed step-sizes are used, then we may still have convergence, but now the iterates "oscillate" about their limiting values with variance proportional to the step-size. We also initialise $\theta^{r,j} = 0$ for all $(r, j) \in \mathcal{E}$. In Figure 3 we plot the errors $\theta_*^{r,j} - \theta_n^{r,j}$ for each edge's relative coordinate estimate against iteration n. We see that all errors converge to zero.

As done for the RML, we present an EM solution for the linear Gaussian problem, when the sensor network in Figure 1 is used for tracking. Again root nodes are $\{3, 4, 6, 9\}$ and we plot the errors $\theta_*^{r,j} - \theta_n^{r,j}$ for each edge in Figure 4.

5.1 EKF Implementation

We would like to solve the self localization problem using an EKF implementation, when each node performs bearings only tracking for a deployment of sensors as in Figure 1 . At each node r, the target being tracked yields observation $Y_{r,n}$ and obeys the following dynamics

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

with $W_n^r \overset{i.i.d.}{\sim} \mathcal{N}(0, 0.1)$. When linearisation is used and the root nodes are $\{3, 4, 6, 9\}$, we plot the errors $\theta_*^{r,j} - \theta_n^{r,j}$ for each edge in Figure 5.

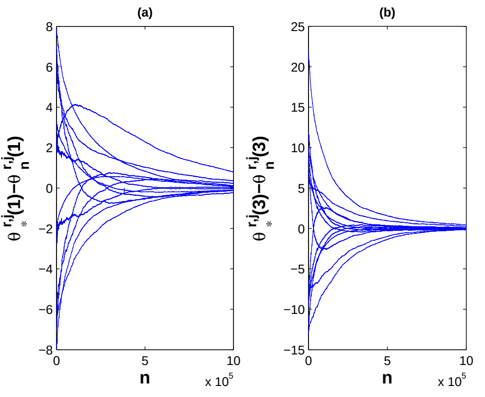


Figure 5. Error at each EKF iteration between the true parameter and current update of the localisation parameter for all edges of the sensor net. (a) and (b) show the errors in the x- and y- coordinates respectively.

6. Conclusions

In this paper, we have presented a general framework to perform recursive static parameter estimation in dynamic graphical models. We derive fully decentralized algorithms using RML and EM to solve the sensor localisation problem. For linear Gaussian graphs, our algorithm can be implemented exactly using a distributed version of the Kalman filter and its derivative. In the non linear case, an EKF solution has been proposed. A Sequential Monte Carlo algorithm to solve the general nonlinear and non Gaussian problem is under preparation.

References

[1] C. Andrieu, A. Doucet, V.B. Tadic, "Online simulation-

based methods for parameter estimation in non linear non Gaussian state-space models ", Proc. IEEE CDC/ECC, Sevilla, Dec. 2005

[2] I.F. Akyildiz, W. Su , Y. Sankarasubramaniam, E. Cayirci,"A Survey on Sensor Networks ", *IEEE Communications Magazine*, Vol.40(8), pp:102–114, August, 2002

[3] 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, June 18-20, 2007

[4] Y. Bar-Shalom, and X. R. Li, Estimation and Tracking: Principles, Techniques, and Software, Artech House Inc.,1993.

[5] N.P. Demster, N.M. Lair , and D.B. Rubin, "Maximum likelihood from incomplete data via the E.M. algorithm,"Journal of the Royal statistical society of London, Series B 39:1-38. RSSL 1977.

[6] A. Doucet, J.F.G. de Freitas and N.J. Gordon, Sequential Monte Carlo Methods in Practice. New York: Springer, 2001.

[7] S. Funiak, C. Guestrin, M. Pashkin, and R. Sukthankar, "Distributed localization of networked cameras ", Proc. of the fifth international conference on Information processing in sensor networks, Nashville, Tennessee, USA, 2006.

[8] A. T. Ihler and J. W. Fisher and R. L. Moses and A. S. Willsky, "Nonparametric Belief Propagation for Self-Localisation in Sensor Networks, " IPSN'04: Proc. of 3rd IPSN, Berkeley, California, 2004

[9] F. LeGland, and L. Mevel, "Recursive Identification in Hidden Markov Models" Proc. of 36th IEEE CDC, pp:3468-3473, 1997.

[10] R. Nowak, "Distributed EM Algorithms for Density Estimation and Clustering in Sensor Networks " IEEE Transaction of Signal Processing in Networking, August 2003.

[11] N. Okello, and S. Challa, V"Joint Sensor Registration for Distributed Trackers, " *IEEE Transactions on Aerospace and Electronic Systems*, March, 2003.

[12] N. Patwari, J. Ash, S. Kyperountas, A.O. Hero, R.L. Moses, and N.S. Correal, "Locating the nodes, cooperative localisation in wireless sensor networks, " IEEE Signal Processing, Vol. 54, July 2005.

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

[14] N. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Mobile-Assisted Localization in Wireless Sensor Networks ", IEEE INFOCOM, Miami, FL, March 2005.

[15] H. E. Rauch, F. Tung, and C. T. Striebel, "Maximum likelihood estimates of linear dynamic systems, "J. Amer. Inst Aeronautics and Astronautics 3 (8) (1965) 1445–1450.

[16] C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe, and A. Grue, "Simultaneous localization, calibration, and tracking in an ad hoc sensor network ",In Proc. IPSN, Nashville, Tennessee, USA, 2006.

[17] J. Vermaak, S. Maskell, and M. Briers, "Online Sensor Registration, " IEEE Aerospace Conference, Big Sky, MT, 2006 (to appear).

[18] Xiao L., Boyd S. and Lall S., "A Scheme for Robust Distributed Sensor Fusion Based on Average Consensus, " *IPSN'05*, April 2005, pp:63-70.