Automatica 95 (2018) 63–72

Contents lists available at ScienceDirect

Automatica

journal homepage: www.elsevier.com/locate/automatica

A distributed Kalman filtering algorithm with fast finite-time convergence for sensor networks☆

Zongze Wu [a], Minyue Fu [b,a,*], Yong Xu [a], Renquan Lu [a]

[a] School of Automation, Guangdong University of Technology, and Guangdong Key Laboratory of IoT Information Technology, Guangzhou 510006, China
[b] School of Electrical Engineering and Computer Science, The University of Newcastle, NSW 2308, Australia

ARTICLE INFO

Article history:
Received 24 January 2017
Received in revised form 9 December 2017
Accepted 17 April 2018
Available online 29 May 2018

Keywords:
Distributed estimation
Distributed Kalman filtering
Sensor networks
Maximum likelihood estimation
Weighted least-squares estimation
Distributed field estimation

ABSTRACT

This paper proposes a new distributed algorithm for Kalman filtering. It is assumed that a linear discrete-time dynamic system is monitored by a network of sensors with some being active and some idle. The goal of distributed state estimation is to devise a distributed algorithm such that each node can independently compute the optimal state estimate by using its local measurements and information exchange with its neighbours. The proposed algorithm applies to acyclic network graphs (i.e., tree graphs) with fast finite-time convergence, but is also applicable to cyclic graphs by combining it with a distributed loop removal algorithm. The proposed algorithm enjoys low complexities, robustness against transmission adversaries and asynchronous implementability. The proposed distributed algorithm also applies to maximum likelihood estimation and weighted least-squares estimation, as special cases. With simple modifications, the proposed algorithm also applies to an important problem in signal processing called distributed field estimation.

1. Introduction

Distributed Kalman filtering (DKF) has been an extremely active research topic for over a decade, in response to rapid development and vast deployment of low-cost sensors and sensor networks. The technical challenge is how to migrate the well-established central (or traditional) Kalman filtering (KF) approach (Anderson & Moore, 1979; Kalman, 1960) to complex large-scale dynamic systems with measurements distributed over a large geographical area (Khan & Moura, 2008). Available DKF algorithms are already abundant. For example, a one-step prediction algorithm was introduced in Zhou (2013); a distributed iterate-collapse inversion algorithm in conjunction with a bipartite fusion graph was introduced in Khan and Moura (2008) for spatially sparse systems; distributed fusion estimation was proposed in Chen, Zhang, and Yu (2014a, b) and Chen, Zhang, Yu, Hu, and Song (2015); DKF using quantized information were studied in Li, Kar, Alsaadi, Dobaie, and Cui (2015), Riberio, Giannakis, and Roumeliotis (2006) and Song, Yu, and Zhang (2014); a DKF design using the well-known gossip protocol was given in Li, Kar, Moura, Poor, and Cui (2015); DKF using diffusion strategies was studied in Cattivelli and Sayed (2010) and Hu, Xie, and Zhang (2012); DKF with out-of-sequence measurements was treated in Shen, Song, Zhu, and Luo (2009), and fusion-centre based DKF designs were shown in Song, Xu, and Zhu (2014) and Xu, Song, Luo, and Zhu (2012). See Mahmoud and Khalid (2013) for a recent survey on DKF. Related works also include distributed maximum likelihood estimation (Zhao & Nehorai, 2007) and distributed weighted least-squares estimation (Marelli & Fu, 2015).

Numerous applications of DKF have been reported in the literature, ranging from environmental monitoring to surveillance, detection, tracking and object classification. Target tracking using a sensor network over a large geographical area is an active research topic recently, and DKF has been shown to play an important role in this application (Medeiros, Park, & Kak, 2008; Zhou, Fang, & Hong, 2013). In fact, this line of research can be traced back to Durrant-Whyte and Rao (1991) in 1991 and Regazzoni (1994) in 1994. DKF also finds wide applications in plant-wide control systems (Vadigepalli & Doyle, 2003), stochastic nonlinear systems with communication delays and packet losses (Wang, Fang, & Liu, 2015), clock synchronization for sensor networks (Luo & Wu, 2013), wireless sensor networks (Riberio & Giannakis, 2006), spatial estimation (Cortés, 2009), power networks (Kanna, Dini, Xia, Hui, & Mandic, 2015; Roshany-Yamchi et al., 2013; Sun, Fu, Wang, & Zhang, 2015; Tai, Lin, Fu, & Sun, 2013). We will also show in this paper a novel

☆ The material in this paper was not presented at any conference. This paper was recommended for publication in revised form by Associate Editor Bert Tanner under the direction of Editor Christos G. Cassandras.

* Corresponding author at: School of Electrical Engineering and Computer Science, The University of Newcastle, NSW 2308, Australia.
E-mail addresses: zzwu@scut.edu.cn (Z. Wu), minyue.fu@newcastle.edu.au (M. Fu), yxu@hdu.edu.cn (Y. Xu), rqlu@gdut.edu.cn (R. Lu).

https://doi.org/10.1016/j.automatica.2018.05.012
0005-1098/

application of DKF in distributed field estimation where a sensor network is used to estimate the parameters of a physical field over a large geographical area, a problem with vast applications on its own (see, e.g., Martinez, 2010; Talarico, Schmid, Alkhweldi, & Valenti, 2014; Wang, Ishwar, & Saligrama, 2008).

One common approach to DKF is to use an average consensus strategy as introduced in Xiao and Boyd (2004); see Carli, Chiuso, Schenato, and Zampieri (2008), Das and Moura (2015), Kar and Moura (2011), Song, Yu et al. (2014), Xu et al. (2012), and Zhou et al. (2013) for examples of this approach. The main shortcomings of this approach include (1) usually only asymptotic convergence is guaranteed, meaning that an infinite number of iterations is required in theory; (2) stopping criteria are difficult to give for practical applications; (3) only sub-optimal estimates are usually given. Another common problem with many existing DKF algorithms is that a fusion centre is required, which means that they are not fully distributed. Approaches of this kind include Chen et al. (2014a, b, 2015), Shen et al. (2009), Song, Xu et al. (2014) and Xu et al. (2012). We also note that fusion centre is also commonly used in the so-called parallel Kalman filtering where local measurements are used to produce local estimates that are then fused together in a fusion centre; see, e.g., Hashemipour, Sumit, and Laub (1988). The need for fully distributed DKF with good optimality properties and low computational, communicational and storage complexities is urgent. These features are essential to make DKF scalable to large-scale sensor networks.

In this paper, we consider a sensor network used to detect, monitor and track targets within the geographical area covered by the network. Each target is modelled as a linear dynamic system, suitable for describing the motion of a moving target or changes in time-varying parameters. The goal of this paper is to devise a distributed algorithm that allows us to estimate the state of each target system. The algorithm needs to have low complexities per node (in terms of communication, computation and storage) so that it is *scalable* to large-scale sensor networks. For convenience, we mainly consider the tracking of a single target system, as the tracking of multiple targets can be achieved by a multiple number of single target trackers. We consider the scenario where the target system is measured by a small subset of active sensors at each time instant, while the rest of the sensors are idle. The set of active sensors is allowed to vary for each time. The use of idle nodes is motivated by applications where only a small fraction of sensors are able to measure information for a particular system. For example, a surveillance network may be responsible to monitor certain types of targets over a large geographical area. A moving target may be observed only by a small number of sensors around it, but the information about it may need to be distributed among the whole network so that the target can be tracked as it traverses within the network.

Following the standard (central) KF approach, we also divide the DKF problem into two steps, a maximum likelihood estimation (MLE) step and a one-step forward prediction step. The core technical issue is how to carry out the MLE step in a fully distributed manner. For this, we propose a fully distributed maximum likelihood estimation (DMLE) algorithm. Under the assumption that the communication graph for the sensor network is acyclic (i.e. it is a tree graph), the algorithm delivers the same (optimal) estimate as given by a central MLE algorithm, but with the advantage of fast convergence. That is, the proposed DMLE algorithm converges in a finite number of iterations (this number equals the maximum diameter of the graph). For sensor networks with a cyclic graph, we can apply a distributed depth-first-search (DFS) algorithm to convert the given graph to a spanning tree before applying the DMLE algorithm. We will show that the proposed DKF algorithm, which is based on the aforementioned DMLE algorithm, enjoys a number of nice properties, including low computational, communicational and storage complexities, robustness against transmission loss and delay and asynchronous implementability. The algorithm can run in either a point-to-point communication mode between neighbouring sensors (for better privacy) or in a broadcast mode (for lower communication burden).

We emphasize that the above target tracking setting is used to motivate the proposed DKF algorithm. Other applications of the proposed DKF algorithm include cascading failure monitoring in a power network, vehicle tracking in a transportation network, fire monitoring in a forest, etc. In particular, we will study the application to distributed field estimation in detail. To help illustrate the proposed DKF, two simulation examples will be given, one on single and multiple target tracking and one on distributed field estimation.

The rest of the paper is organized as follows: Section 2 formulates the DKF problem; Section 3 gives the proposed DKF algorithm; Section 4 details a number of properties of this algorithm; Section 5 discusses its application to distributed field estimation; Section 6 demonstrates the algorithm via some simulation examples; and Section 7 concludes the paper.

2. Problem formulation

Consider a dynamic model for a target system described by

$$x(t+1) = A(t)x(t) + w(t), \tag{1}$$

where $t = 0, 1, \ldots$ is the time index, $x(t) \in \mathcal{R}^n$ is the state, $w(t) \in \mathcal{R}^n$ is a zero-mean i.i.d. Gaussian noise with covariance $W(t) \geq 0$, and $A(t) \in \mathcal{R}^{n\times n}$ is the (possibly) time-varying transition matrix. The initial state $x(0)$ is an independent Gaussian variable with mean x_0 and covariance $P_0 > 0$.

The target system is measured by a network of sensors which can be represented by a graph $\mathcal{G}(t) = \{\mathcal{V}(t), \mathcal{E}(t)\}$ with a set of nodes $\mathcal{V}(t)$ and a set of edges $\mathcal{E}(t) \subset \{(i, j) : i \neq j, i, j \in \mathcal{V}(t)\}$. The set $\mathcal{V}(t) = \mathcal{I}(t) \cup \mathcal{S}(t)$, where $\mathcal{S}(t)$ is a set of sensing nodes with measurements and $\mathcal{I}(t)$ is an idle set consisting of nodes without measurements. Each node $i \in \mathcal{S}(t)$ has a measurement:

$$y_i(t) = C_i(t)x(t) + v_i(t), \tag{2}$$

where $y_i(t) \in \mathcal{R}^{r_i}$, $C_i(t) \in \mathcal{R}^{r_i\times n}$ is the (possibly) time-varying measurement vector, $v_i(t)$ is the measurement noise which is a zero-mean i.i.d. Gaussian noise with covariance $R_i(t) > 0$. Stacking up all the measurements, we get

$$y(t) = C(t)x(t) + v(t). \tag{3}$$

The covariance of $v(t)$ is $R(t) = \text{diag}\{R_i(t)\}$. It is assumed throughout the paper that the system with (1) and (3) is observable.

We assume that $\mathcal{G}(t)$ is undirected and acyclic. A graph is called *undirected* if each edge is undirected. A graph is called *acyclic* if it is *connected* and has no loops, i.e., it is a *tree graph*. We will show how to deal with cyclic graphs later. Denote by $\mathcal{N}_i(t)$ the set of neighbouring nodes connected to node i, and denote by $|\mathcal{N}_i(t)|$ the cardinality of $\mathcal{N}_i(t)$. We assume that $|\mathcal{N}_i(t)| \ll |\mathcal{V}(t)|$ for each $i \in \mathcal{V}(t)$. We denote by $d(t)$ the diameter of the graph $\mathcal{G}(t)$, which is the length of the longest path between two nodes.

For notational convenience, we will suppress the time dependence of the system parameters ($A(t)$, $C(t)$, $W(t)$, $R(t)$) as well as those of the network graph ($\mathcal{G}(t)$, $\mathcal{V}(t)$, $\mathcal{I}(t)$, $\mathcal{S}(t)$, $\mathcal{E}(t)$, $\mathcal{N}_i(t)$, $d(t)$). In addition, with some abuse of notation, $i \in \mathcal{G}$ means $\in \mathcal{V}$, and $(i, j) \in \mathcal{G}$ means $(i, j) \in \mathcal{E}$.

It is well known (Anderson & Moore, 1979) that the optimal state estimation, in the maximum likelihood sense, is given by the celebrated Kalman filter (KF), which we will call *central Kalman filter*. Given the system model (1) and the measurement model

(3), the optimal state estimate $\hat{x}(t+1)$ of $x(t+1)$, conditioned on measurements $y(0), y(1), \ldots, y(t)$, is given by

$$\hat{x}(t+1) = A\hat{x}(t) + AG_t(y(t) - C\hat{x}(t)) \tag{4}$$

with

$$G_t = P_t C^T(t)(CP_tC^T + R)^{-1}, \tag{5}$$

where P_t is the estimation error covariance for $\hat{x}(t)$ given by

$$P_{t+1} = AP_tA^T + W - AP_tC^T(CP_tC^T + R)^{-1}CP_tA^T. \tag{6}$$

An alternative expression for the above is as follows (which we will use for developing a distributed estimation algorithm):

- Update:

$$\tilde{x}(t) = \hat{x}(t) + G_t(y(t) - C\hat{x}(t)); \tag{7}$$

$$\tilde{P}_t = P_t - P_tC^T(CP_tC^T + R)^{-1}CP_t. \tag{8}$$

- Prediction:

$$\hat{x}(t+1) = A\tilde{x}(t); \tag{9}$$

$$P_{t+1} = A\tilde{P}_tA^T + W. \tag{10}$$

The problem of *distributed Kalman filtering* (DKF) is to devise an iterative algorithm which runs on every node $i \in \mathcal{V}_t$ to compute an estimate of $x(t)$ such that after a certain number of iterations, each node's estimate, denoted by $\hat{x}_i(t)$, will converge to the optimal state estimate $\hat{x}(t)$ given by the central KF.

Before ending this section, we need to clarify what we mean by "distributed". In this paper, an algorithm is called *distributed* if the following *linear complexity constraints* on communication, computation and storage are satisfied:

(1) Local information exchange: Each node i can exchange information with each $j \in \mathcal{N}_i$ only once per iteration.
(2) Local computation: Each node i's computational load should be at most $O(|\mathcal{N}_i|)$ per iteration.
(3) Local storage: Each node i's storage should be at most $O(|\mathcal{N}_i|)$ over all iterations.

3. Distributed algorithm for Kalman filtering

In this section, we introduce our DKF algorithm, provide its key property on acyclic graphs and analyse its complexities.

3.1. Distributed maximum likelihood estimation

The main difficulty for distributed implementation of the KF lies in the Update step (7)–(8). This step is also known as the *maximum likelihood estimation* (Anderson & Moore, 1979). The problem is to produce the posteriori estimate $\tilde{x}(t)$ of $x(t)$, given its prior estimate $\hat{x}(t)$ and measurement $y(t)$, i.e., we need to compute the following conditional expectation (see Anderson & Moore, 1979):

$$\tilde{x}(t) = \mathcal{E}\{x(t)|\hat{x}(t), y(t)\}. \tag{11}$$

Its solution is, of course, given by (7)–(8); see Anderson and Moore (1979).

The proposed distributed maximum likelihood estimation algorithm is given in Algorithm 1. It is assumed that the graph $\mathcal{G}$ is acyclic with diameter d.

Theorem 1. *Under the assumption that $\mathcal{G}$ is acyclic with diameter d, the estimate $\tilde{x}(t)$ and the associated error covariance $\tilde{P}_t$ computed by Algorithm 1 coincide with those by (7)–(8).*

Algorithm 1 (Distributed Maximum Likelihood Estimation)

Initialization: For each node i and each $j \in \mathcal{N}_i$,

- if $i \in \mathcal{S}$, set $Q_{i\to j}(0) = C_i^T R_i^{-1} C_i$ and $\alpha_{i\to j}(0) = C_i^T R_i^{-1} y_i(t)$,
- else ($i \in \mathcal{I}$), set $Q_{i\to j}(0) = 0$ and $\alpha_{i\to j}(0) = 0$;
- then transmit the above to node j.

Main loop: At iteration $k = 1, 2, \cdots, d$, for each node i,

- if $i \in \mathcal{S}$, compute

$$Q_i(k) = C_i^T R_i^{-1} C_i + \sum_{j\in\mathcal{N}_i} Q_{j\to i}(k-1); \tag{12}$$

$$\alpha_i(k) = C_i^T R_i^{-1} y_i(t) + \sum_{j\in\mathcal{N}_i} \alpha_{j\to i}(k-1), \tag{13}$$

- else ($i \in \mathcal{I}$), comptue

$$Q_i(k) = \sum_{j\in\mathcal{N}_i} Q_{j\to i}(k-1); \tag{14}$$

$$\alpha_i(k) = \sum_{j\in\mathcal{N}_i} \alpha_{j\to i}(k-1), \tag{15}$$

- then for each $j \in \mathcal{N}_i$,

$$Q_{i\to j}(k) = Q_i(k) - Q_{j\to i}(k-1); \tag{16}$$

$$\alpha_{i\to j}(k) = \alpha_i(k) - \alpha_{j\to i}(k-1), \tag{17}$$

and transmit them to node j.

Termination:

$$\tilde{P}_t = (P_t^{-1} + Q_i(d))^{-1}; \tag{18}$$

$$\tilde{x}(t) = \tilde{P}_t(P_t^{-1}\hat{x}(t) + \alpha_i(d)). \tag{19}$$

Proof. In the sequel, for a given state estimate $\hat{x}$ and the associated estimation error covariance $P > 0$, we will denote the *information matrix* as $Q = P^{-1}$ and the *scaled state estimate* as $\alpha = Q\hat{x}$.

Using the well-known matrix inverse lemma, (8) can be rewritten as

$$\tilde{P}_t^{-1} = P_t^{-1} + C^T R^{-1} C.$$

In view of the composition of C and the fact that R is a block diagonal matrix, the above can be further rewritten as

$$\tilde{P}_t^{-1} = P_t^{-1} + \sum_{i\in\mathcal{S}} C_i^T R_i^{-1} C_i. \tag{20}$$

It is also straightforward to verify that (7) can be rewritten as

$$\tilde{x}(t) = \tilde{P}_t(P_t^{-1}\hat{x}(t) + C^T R^{-1} y(t)) = \tilde{P}_t(P_t^{-1}\hat{x}(t) + \sum_{i\in\mathcal{S}} C_i^T R_i^{-1} y_i(t)). \tag{21}$$

A similar expression of the above can be traced back to Hashemipour et al. (1988).

Take any edge $(i, j) \in \mathcal{E}$. Because $\mathcal{G}$ is a tree graph, $\mathcal{G}$ can be split into two sub-graphs $\mathcal{G}_i$ and $\mathcal{G}_j$ with a connecting edge (i, j) between them. Here $\mathcal{G}_i$ is a tree graph with node i as the root node, and $\mathcal{G}_j$ is a tree graph with node j as the root node, and they are obtained from $\mathcal{G}$ by removing the edge (i, j). For each node in $\mathcal{G}_i$, denote its layer by the number of hops it is away from node i. That is, node i is called the layer-0 node; all the nodes one hop away from node i are called layer-1 nodes, those two hops away are called

layer-2 nodes, and so on. It is clear that the initialization steps set $Q_{i\to j}(0)$ as the information matrix and $\alpha_{i\to j}(0)$ as the scaled state estimate for $x(t)$ using the layer-0 information in $\mathcal{G}_i$ only. Then, tracing through (12)–(17) will lead us to see that after iteration 1, $Q_{i\to j}(1)$ will be the information matrix for $x(t)$ using layer 0 and layer 1 information in $\mathcal{G}_i$ only, and no information from $\mathcal{G}_j$ is used. Similarly, $\alpha_{i\to j}(1)$ will be the scaled state estimate for $x(t)$ using layer 0 and layer 1 information in $\mathcal{G}_i$ only, without using information from $\mathcal{G}_j$. This process goes on. So in iteration d, $Q_{i\to j}(d)$ and $\alpha_{i\to j}(d)$ will be the information matrix and the scaled state estimate for $x(t)$, respectively, using all the information from layer 0 to layer d in $\mathcal{G}_i$, without using information from $\mathcal{G}_j$. Similarly, $Q_{j\to i}(d-1)$ and $\alpha_{j\to i}(d-1)$ will be the information matrix and the scaled state estimate for $x(t)$, respectively, using all the information from layer 0 to layer $d-1$ in $\mathcal{G}_j$, without using information from $\mathcal{G}_i$. We combine them together (using (16)–(17)) to create

$$Q_i(d) = Q_{i\to j}(d) + Q_{j\to i}(d-1)$$
$$\alpha_i(d) = \alpha_{i\to j}(d) + \alpha_{j\to i}(d-1),$$

and note that node j is one hop away from node j and that the diameter of $\mathcal{G}$ is d. Then, it is clear that $Q_i(d)$ and $\alpha_i(d)$ will be the information matrix and the scaled state estimate for $x(t)$, respectively, using all the nodes in $\mathcal{G}$. In particular, these results are independent of node i. Finally, (18) combines the prior information matrixP_t^{-1} with $Q_i(d)$ to create the posterior information matrix $\tilde{P}_t$. Similarly, (19) combines the prior scaled state estimate $P_t^{-1}\hat{x}(t)$ with $\alpha_i(d)$ to create the posterior scaled state estimate $P_t^{-1}\hat{x}(t) + \alpha_i(d)$ and then the posterior state estimate $\tilde{x}(t)$.

Remark 2. We give some interpretations for Algorithm 1 . Following the proof of Theorem 1, we see that the Initialization part sets the scaled state estimate $\alpha_{i\to j}(0)$ and the associated information matrix $Q_{i\to j}(0)$ for $x(t)$ and allows them to be broadcast to all of the neighbouring nodes of i. The steps (12)–(15) *fuse* the scaled estates and the information matrices together. For $i \in \mathcal{I}$, only the neighbouring information needs to be fused together; whereas for $i \in \mathcal{S}$, the measurement y_i needs to be added as well. The steps (16)–(17) ensure that the information transmitted back to node j will not be "contaminated" by the information coming from node j previously. The avoidance of contamination marks a crucial difference between the proposed distributed maximum likelihood estimation algorithm and many distributed algorithms in the literature, and is the key for finite-time convergence.

Remark 3. We see from the proof of Theorem 1 that the computation of $\tilde{P}_t^{-1}$ in (20) involves the fusion of the terms $C_i^T R_i^{-1} C_i$ for all $i \in \mathcal{S}$. Similarly, the fusion of $C_i^T R_i^{-1} y_i(t)$ is needed for computing $\tilde{x}(t)$ in (21). Since the fusion is done iteratively, it is natural to ask what the minimum number of iterations is required to do the above computations. The following result gives the answer, which confirms that Algorithm 1 achieves the fastest possible finite-time convergence for acyclic graphs.

Lemma 4. *Under the linear complexity constraint (1) in* Section 2*, a connected undirected graph $\mathcal{G}$ with diameter d needs a minimum of d iterations to achieve the fusion* (20) *or* (21)*.*

Proof. Let nodes i and j be such that they are d hops away from each other (such nodes exist by the definition of d). The information at node i needs to propagate to node j in order for node j to correctly fuse the $C_i^T R_i^{-1} C_i$ (or $C_i^T R_i^{-1} y_i(t)$) term. By the linear complexity constraint (1), this will take at least d iterations.

Remark 5. Since our distributed estimate and its error covariance coincide with that of the central Kalman filter, our distributed Kalman filter is stable due to the well-known fact that the central Kalman filter is stable (Anderson & Moore, 1979).

3.2. Distributed prediction

The distributed implementation of the Prediction step (9)–(10) is easy, given that each node $i \in \mathcal{V}$ has a local estimate equal to $\tilde{x}(t)$ and the corresponding estimation error covariance equal to $\tilde{P}_t$. That is, each node $i \in \mathcal{V}$ computes (9)–(10) independently.

3.3. Application to cyclic network graphs

A main restriction of the proposed algorithm for DKF is that the measurement network graph needs to be acyclic. For a cyclic graph, it is necessary to prune loop-forming edges so that the remaining graph becomes loop free. The resulting graph is called a *spanning tree*, a tree graph with all the nodes in the original graph.

Constructing a spanning tree is a well-known classical problem in graph theory. The standard (and most popular) algorithms are *depth-first search* (DFS) and *breath-first search* (BFS) (Goodrich & Tamassia, 2001; Skiena, 2008). Both algorithms can construct a spanning tree in $O(|\mathcal{V}| + |\mathcal{E}|)$ time. However, these algorithms are sequential, i.e., not distributed. Indeed, both algorithms start with an arbitrarily chosen root node. In DFS, one starts at the root and explores as far as possible along each branch before backtracking and exploring another branch; whereas in BFS, one starts at the root and explores the neighbouring nodes first, before moving to the next level neighbours.

In our recent work (Xie & Fu, 2018), we proposed a distributed DFS algorithm (Algorithm 2) for constructing a spanning tree. The algorithm applies to a connected graph with diameter d, with the assumption that each node in the graph has a distinct numerical ID number.

Algorithm 2 (Distributed DFS Algorithm for Spanning Tree)

Initialization: Select a root node in $\mathcal{G}$ (by running max-consensus d iterations and setting the node with maximum ID number as the root node). Mark the root node as "visited" and and all other nodes as "unvisited". Transmit a token from the root node to each of its neighbouring nodes.

Iterations $k = 1, 2, \ldots$: For each node i, do the following:

(1) If it does not receive the token, do nothing;

(2) If it is "unvisited" and it receives the token from only *one* neighbour, then mark the node as "visited", and relay the token to all other neighbouring nodes without the "removal" mark, except the incoming edge (i.e., the edge where the token came from);

(3) If it is "unvisited" and it receives the token from *multiple* neighbours, then mark the node as "visited", leave one (any one) incoming edge alone and mark all other the incoming edges as "removal", and then relay the token to all other neighbouring nodes without the "removal" mark, except the remaining incoming edge;

(4) If it is already "visited" and it receives the token, mark the incoming edge as "removal" and do not relay the token further.

Algorithm 2 uses a max-consensus algorithm (Nejad, Attia, & Raisch, 2009), which is a distributed algorithm for finding the maximum value in a connected graph. For a graph with diameter d and n nodes (each node i with a variable a_i), the algorithm converges in d iterations. For completeness, the algorithm is described below:

- Each node i initializes $a_i(0) = a_i$ and transmits it to its neighbours.

- For iterations $k = 1, 2, \ldots$, each node i updates $a_i(k) = \max_{j\in\mathcal{N}_i} a_j(k-1)$ and transmits it to its neighbours.

As shown in Xie and Fu (2018), Algorithm 2 takes at most $d+1$ iterations to finish. Moreover, it has very low complexity for each node because, during each iteration, each node relays the token only to its neighbours which were not visited before.

4. Properties

In this section, we explore the properties, other than the finite convergence property stated in Theorem 1. These properties include robustness against transmission adversaries, asynchronous implementability, and broadcast transmission mode. We will also discuss a special case of Algorithm 1 for distributed weighted least-squares estimation and application of Algorithm 1 to multi-target estimation.

4.1. Robustness against transmission loss and delay

It turns out that Algorithm 1 is very resilient to transmission loss and delay due to the fact that each node uses the most recently received information from its neighbours to update its own information. A delayed arrival of information from neighbours will only delay the update and the convergence of the algorithm, without affecting the maximum likelihood estimates. Likewise, if packet loss happens, it is sufficient that retransmission happens and eventually the information arrives, which can be guaranteed by most communications protocols, e.g., Transmission Control Protocol (TCP), via Acknowledgement (ACK). This will result in transmission delay only and will not affect the maximum likelihood estimates. The only requirement to ensure the correctness of the DKF result is that the delayed information needs to arrive within one sampling period of the system.

4.2. Asynchronous implementation

Although Algorithm 1 appears as a synchronous algorithm (i.e., all the nodes update together at each iteration time k), this is purely for convenience. The algorithm can be implemented asynchronously as follows: Each node waits till it receives new information from its neighbours, then updates its own information and transmits it to its neighbours. In fact, each node can either update information as soon as it receives some update from neighbours, or wait till multiple updates arrive or till a prescribed time interval elapses. The only tradeoff is that more frequent updates require more transmissions and less frequent updates cause more delay. As discussed earlier, as long as the all the iterations finish within one sampling period, no degradation will occur to the DKF result.

4.3. Point-to-point transmission vs. broadcast

Algorithm 1 is written for point-to-point transmissions between neighbours. That is, each node i transmits to each neighbouring node j different information, i.e., $(Q_{i\to j}(k), \alpha_{i\to j}(k))$. Point-to-point transmission has the advantage of good privacy, but at the cost of more transmissions (one for each neighbour). An alternative is broadcast. Algorithm 1 can be easily implemented in a broadcast mode. To do so, each node i still needs to compute $(Q_{i\to j}(k), \alpha_{i\to j}(k))$ for each neighbouring node j, but simply broadcast $(Q_i(k), \alpha_i(k))$ to all of its neighbours. When node j receives $(Q_i(k), \alpha_i(k))$, it can re-compute $(Q_{i\to j}(k), \alpha_{i\to j}(k))$ by subtracting its own $(Q_{j\to i}(k-1), \alpha_{j\to i}(k-1))$. For applications like wireless sensor networks where transmission power and bandwidth are scarce, broadcasting is no doubt a better choice.

4.4. Distributed weighted least-squares estimation

Weighted least-squares estimation is no doubt one of the most popular estimation methods, applicable to a wide range of applications. Technically speaking, this is a special case of maximum likelihood estimation. The only difference is that the prior information is not available for weighted least-squares estimation. This is the same as taking $P_t = \infty$. That is, only the Termination step in Algorithm 1 needs to be revised to

$$\tilde{P}_t = Q_i(d)^{-1}; \tag{22}$$

$$\tilde{x}(t) = \tilde{P}_t \alpha_i(d). \tag{23}$$

4.5. Distributed estimation for multiple targets

The proposed distributed algorithm is readily applicable to track multiple targets. We modify the target model (1) to a set of target models:

$$x^{(m)}(t+1) = A^{(m)}x^{(m)}(t) + w^{(m)}(t),\ m = 1, 2, \ldots, M. \tag{24}$$

For each target m, there may be a set of sensors $\mathcal{S}^{(m)}$ monitoring it, i.e., for each $i \in \mathcal{S}^{(m)}$,

$$y_i^{(m)}(t) = C_i^{(m)}x^{(m)}(t) + v^{(m)}(t). \tag{25}$$

We can either run Algorithm 1 for M times in parallel, one for each target model, or amalgamate the targets into a super-target and then run Algorithm 1 on it. The two implementations may yield different communication complexities depending on whether transmissions for different targets are synchronized or not, and whether they can be packed together.

5. Application to distributed field estimation

In this section, we discuss how the proposed DKF algorithm can be easily modified to solve the distributed field estimation problem.

To illustrate the distributed field estimation problem, consider monitoring the temperature distribution over a large geographical area using a network of temperature sensors. Each sensor measures the temperature at its local point, with unavoidable measurement noise. To reduce the noise effect, all the measurements over a region centred around a sensor are weight-averaged. To further improve the estimation accuracy, a dynamic temperature model is incorporated to provide prior prediction of the temperature change, which can be developed using historical data (e.g., data over past 24 h) or forecast information. Each sensor may have its local temperature model, specialized to its local physical environment. The prior prediction is then calibrated using the averaged measurements. Although we have used temperature distribution monitoring as an example, the same distributed field estimation problem occurs in vast applications, including air pollutant monitoring, water quality monitoring, flow distribution monitoring, traffic congestion monitoring, data traffic monitoring, and so on.

We now formally describe our distributed field estimation problem. Given a sensor network represented by a connected undirected graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ as before, denote by $\mathcal{G}_i(k)$ a sub-graph of $\mathcal{G}$ centred around node i such that all the nodes in it are at most k hops away from node i. Each node i is associated with a state variable $x_i(t) \in \mathcal{R}^m$ of common interest to all nodes (such as the temperature in the example above) at time t, and a linear dynamic model:

$$x_i(t+1) = A_i x_i(t) + w_i(t), \tag{26}$$

where $w_i(t)$ is a zero-mean i.i.d. Gaussian noise with covariance $W_i \geq 0$, and $x_i(0)$ is an independent Gaussian variable with mean

$x_{i,0}$ and covariance $P_{i,0} > 0$. Each node i also has a measurement similar to (2), i.e.,

$$y_i(t) = C_i x_i(t) + v_i(t). \tag{27}$$

Also given is an integer $r \leq d$ describing the size of the geographical region of interest, i.e., each node i is interested only in the subnetwork associated with $\mathcal{G}_i(r)$.

In our distributed field estimation problem, we recognize the fact that a field distribution tends to have high correlations among neighbouring nodes. Instead of modelling the correlations explicitly (which is not always easy), we take the view that each node i regards all the nodes in $\mathcal{G}_i(r)$ to have roughly the same state, i.e., $x_j(t) \approx x_i(t)$ for all $j \in \mathcal{G}_r$. This means that, as far as node i is concerned, there is a common system model for all the nodes in $\mathcal{G}_i(r)$, which is (26), and the measurement $y_j(t)$ available at $j \in \mathcal{G}_i(r)$ is

$$y_j(t) \approx C_j x_i(t) + v_j(t). \tag{28}$$

Due to the above approximation, at each time step t, our distributed field estimation problem involves doing the following for each node i:

(1) Compute the *regional maximum likelihood state estimate*

$$\tilde{x}_i(t) = \mathcal{E}\{x(t)|\hat{x}_i(t), y_j(t), \forall j \in \mathcal{G}_i(r)\}$$

and the associated estimation error covariance $\tilde{P}_i(t)$ using the approximation (28).

(2) Compute, using the above result and (26), the state estimate $\hat{x}_i(t+1)$ for $x_i(t+1)$ and the associated estimation error covariance $P_i(t+1)$.

It is clear that when the region of measurements used for the first step is the entire sensor network and the state variables for different nodes are the same, this problem is the same as the DKF problem we have studied so far. The challenge is, of course, how to modify the proposed algorithm to solve this new problem.

It turns out the required modification is very simple. Firstly, instead of running Algorithm 1 for d iterations, we stop at r iterations. Secondly, in the Termination step, P_t, $\tilde{P}_t$, $\hat{x}(t)$ and $\tilde{x}(t)$ need to be replaced with their local versions by adding the subscript i. What these two changes do is that the maximum likelihood estimate obtained at each node i becomes a regional estimate over $\mathcal{G}_i(r)$. Finally, it is clear that the Prediction step can be easily replaced with its local version. It is interesting to know that the modified algorithm has even lower complexities than the original algorithm due to the lower number of iterations required.

Due to the fact that only r iterations are needed, the graph $\mathcal{G}$ needs not be acyclic. This is because of the nice property of the proposed algorithm that only nodes within r hops away from node i are fused in $\tilde{x}_i(t)$. This means that it is sufficient to have $\mathcal{G}$ void of loops of length r or less. Also note that central Kalman filtering is not directly applicable to the distributed field estimation because each node has its own state estimation problem, i.e. there is no central state to estimate.

6. Simulations

In this section, we provide several simulation examples to demonstrate the proposed DKF algorithm. Our attention will be on the complexities of the algorithm and accuracy of the estimation results. Three examples will be used. The first one is about tracking of a single target in a surveillance network. The second one is a simple generalization of the first one, about tracking of multiple targets. The third one is about distributed field estimation for a temperature field (e.g., in a bushfire monitoring network).

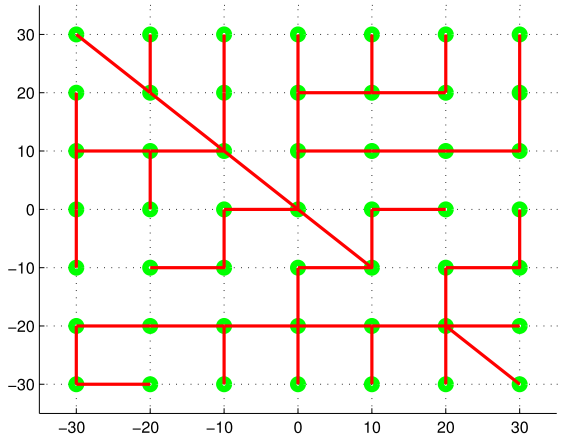

Fig. 1. Sensor network for target tracking.

6.1. Single target tracking

Consider a network of sensors in a two-dimensional region to monitor a possible moving target (e.g., intruder, truck, mobile transmitter). The state of the target is $x(t) = [p^T(t)\ s^T(t)]^T$ with the two-dimensional coordinate $p(t)$ and its velocity $s(t)$. The sampling period is assumed to be T, normalized to be 1 s. Its state-space equation is a random walk model as follows:

$$p(t+1) = p(t) + s(t)T + w_p(t), \tag{29}$$

$$s(t+1) = s(t) + w_s(t). \tag{30}$$

Each active sensor $i \in \mathcal{S}$ measures $p(t)$ with noise, i.e.,

$$y_i(t) = p(t) + v_i(t). \tag{31}$$

We assume that the noises $w(t) = [w_p(t)\ w_s(t)]^T$ and $v_i(t)$ are independent, zero-mean, and their covariances are as follows: $W = \text{diag}\{0.1, 0.05\}$, $R_i = \text{diag}\{0.08, 0.08\}$, $P_0 = \text{diag}\{0.2, 0.2, 0.2, 0.2\}$.

We consider the square monitoring area in Fig. 1, with 7×7 sensors uniformly distributed. It is assumed that each sensor can detect a target around it with the radius of 15. The original communication edges are depicted by the dotted lines., i.e., every node can communicate with any horizontal or vertical node next to it (or, put in another way, any node within the radius of 12). It is obvious that this graph is cyclic. Applying the distributed DFS algorithm results in a spanning tree with the communication edges depicted by the solid lines in Fig. 1.

Fig. 2 shows the target tracking results by the distributed estimator and the central estimator. As expected by Theorem 1, there is no difference between the two.

6.2. Multiple target tracking

We consider the same sensor network in Fig. 1 and add a second target with the following model:

$$A = \begin{bmatrix} 0.9 & 0.9 \\ -0.1 & 0.91 \end{bmatrix},\ C = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The stochastic properties of the initial state and the noises of the measurements and the systems are same as in the first example. The noises for the two targets are independent.

The estimation results are shown in Fig. 3. Again, the results for the distributed estimation and central estimation are the same, as expected.

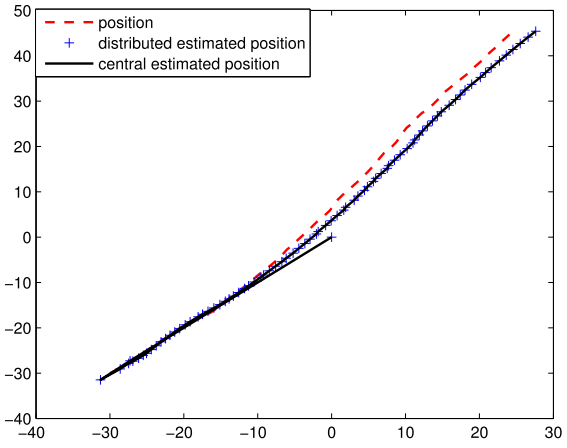

Fig. 2. Single target tracking comparison.

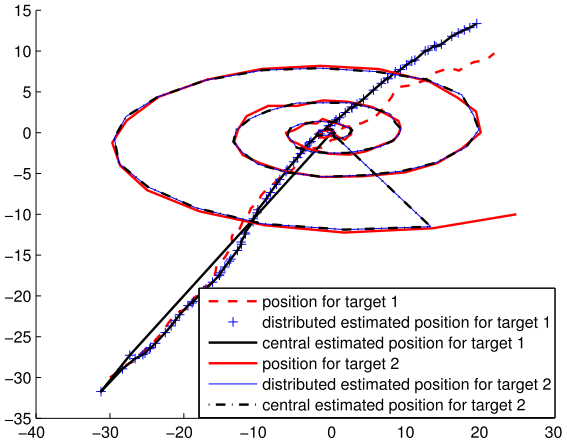

Fig. 3. Multiple target tracking comparison.

6.3. Distributed field estimation

Consider a terrain (e.g., forest, coal mine, agriculture land, chemical field) covered by a wireless sensor network to monitor its temperature distribution. Each sensor is equipped with a temperature sensor. A dynamic temperature model is available at each sensor, which can be obtained from historical data and physical properties of the terrain. It is desirable to smooth out the measurements over a region around each sensor so that measurement noises are reduced and local temperature fluctuations are suppressed. Although it is possible to carry out the required estimation work in a centralized way, this would involve a central processing unit and create communication bottlenecks around it. The example below shows how distributed estimation is done using the proposed algorithm.

A square field of one square kilometres has temperature sensors uniformly distributed, one sensor for every 10×10 square metres. At each node, the temperature $\tau_i(t)$ at each node i and time t is modelled by

$$\tau_i(t) = \tau_0(t) + \delta_i,$$

where $\tau_0(t)$ is the average field temperature at time t, which experiences the following 24-hour cycle (a fictitious model):

$$\tau_0(t) = 10 + 5\sin(\omega t)$$

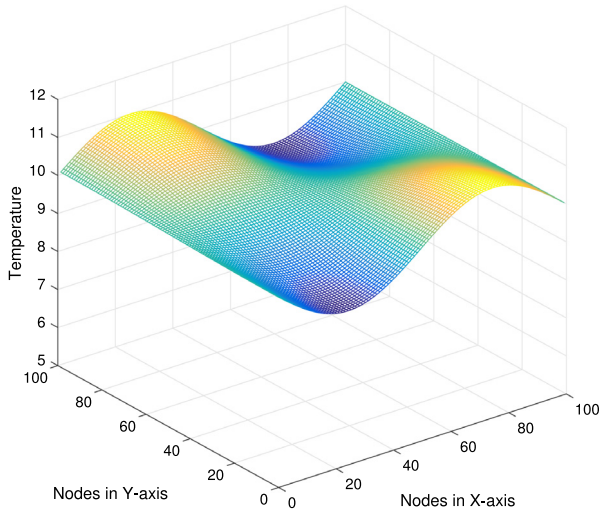

Fig. 4. Temperature distribution.

with $\omega = 2\pi/T_0$ with $T_0 = 24 \times 60$ min, and δ_i represents the local temperature fluctuation. Fig. 4 shows the distribution of $\tau_i(t)$ with $t = 0$ and $\tau_0(0) = 10$ (i.e., Fig. 4 shows the distribution of $\delta_i + 10$). It is assumed in this example that δ_i is time-invariant. The sampling period is chosen to be 1 min. The above can be modelled by

$$x_i(t+1) = \begin{bmatrix} 0 & 1 \\ -1 & 2\cos(\omega) \end{bmatrix} x_i(t)$$

with $x_i(0) = \delta_i + 10$. This gives $\tau_i(t) = [1\ \ 0]x_i(t)$.

The measured temperature at node i is given by

$$y_i(t) = \tau_i(t) + v_i(t) = [1\ \ 0]x_i(t) + v_i(t)$$

with $v_i(t)$ as a Gaussian noise with zero mean and variance of 0.09. The measured temperature distribution at $t = 0$ is shown in Fig. 5.

Choosing $r = 4$ as the size of geographical region of interest and applying the proposed algorithm in Section 5, the resulting estimated temperature distribution is shown in Fig. 6.

6.4. Comparisons

6.4.1. Target tracking examples

We first compare our simulation results for the two target tracking examples with several classes of methods.

Sequential processing method: In a sequential processing method (Zhao & Nehorai, 2007) (although it is also called distributed method in this reference), the maximum likelihood estimation step (or Update step) (7)–(8) is done sequentially. More specifically, nodes $1, 2, \ldots, n$ are arranged in an ordered sequence and initialize $\tilde{x}(t) = \hat{x}(t)$. Then, node 1 updates $\tilde{x}(t)$ using its local measurement $y_1(t)$ and passes the updated $\tilde{x}(t)$ to node 2. This goes on until $\tilde{x}(t)$ is finally updated by node n. Clearly, this sequential processing doe not satisfy our distributed processing properties.

Distributed updating, non-distributed averaging method: In the distributed estimation algorithm in Vadigepalli and Doyle (2003), each node takes the state estimate $\hat{x}_i(k-1|k-1)$ using its own prediction model to obtain $\hat{x}_i(k|k-1)$, then using its local measurement $y_i(k)$ to update its local estimate $\hat{x}_i(k|y_i(k))$, then communicate with all other nodes to obtain an average $\hat{x}_i(k|k)$. Although the result is identical to global estimation, the second step (communications among nodes and averaging) is not done in a distributed fashion. Thus, this does not satisfy our distributed

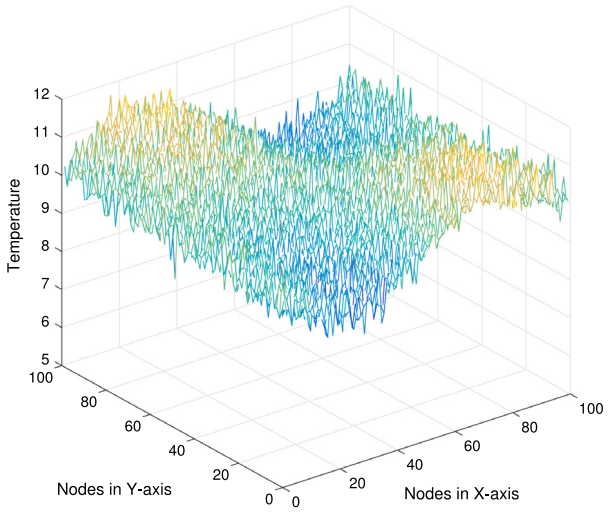

Fig. 5. Measured temperature distribution.

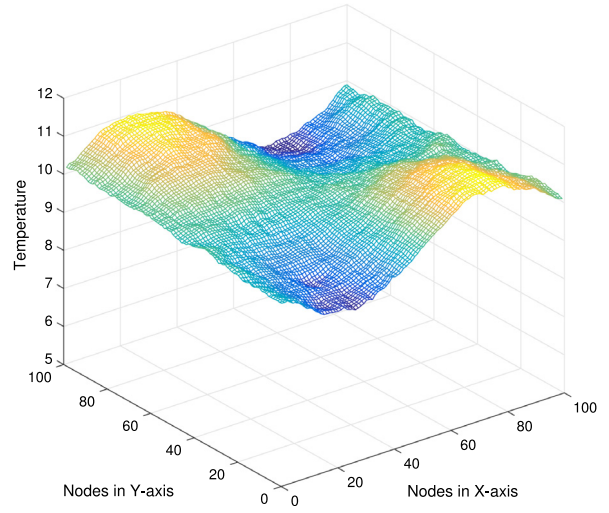

Fig. 6. Estimated temperature distribution.

processing properties. Vadigepalli and Doyle (2003) recognize this problem and suggest to do the averaging exercise at a reduced sampling rate. This would compromise the optimality of the estimation.

Central processor assisted method: Some distributed estimation methods require a fusion centre which collects all the processed local information and fuse them together to produce a global estimate which is then broadcast back to each node; Chen et al. (2014a, b, 2015) and Song, Xu et al. (2014). Obviously these methods are not fully distributed in our sense.

Laplacian matrix-based average consensus method: Many distributed Kalman filtering algorithms are available based on this method, as mentioned in Introduction. Here we discuss Carli et al. (2008) as a representative algorithm. In Carli et al. (2008), each node i takes $\hat{x}(t)$ and computes its own update $\tilde{x}_i(t)$ using its local measurement $y_i(t)$, then uses a Laplacian matrix based average consensus algorithm to produce the final update $\tilde{x}(t)$, which theoretically requires an infinite number of iterations to achieve consensus. Carli et al. (2008) use a finite number of iterations, but this compromises the optimality, and the number of iterations cannot be determined *a priori*. In an example in Carli et al. (2008), a randomly distributed network of 30 nodes needs 10 iterations, but it is not clear how the required number of iterations grows as the network size grows. In contrast, the proposed algorithm needs only d iterations. Other references using the Laplacian matrix based average consensus method include Das and Moura (2015), Kar and Moura (2011), Xu et al. (2012) and Zhou et al. (2013), which suffer from similar deficiencies.

In summary, all the methods above, estimation errors are identical or similar to the distributed algorithm given in this paper, hence we do not show the corresponding simulation results for the tracking examples. (The only case where estimation errors are not identical is the Laplacian matrix-based average consensus method with a finite number of iterations.) The main differences lie in either lack of fully distributed processing properties (Methods 1–3 above) or requiring higher computational and communicational complexities (Methods 2 to 4 above).

6.4.2. Distributed field estimation example

We now do comparison for the distributed field estimation example.

Cluster-based method: In Medeiros et al. (2008), a cluster-based distributed Kalman filtering technique is proposed for target tracking. Nodes which can detect the target at a given time instant forms a cluster. In this cluster, a node is chosen as a cluster head which collects measurements from other nodes and updates the state estimate collectively. This updated state estimate is passed on to the cluster head in the next time instant. It is clear that this cluster coincides with our set of sensing nodes $\mathcal{S}(t)$. However, the cluster-based processing is still not fully distributed in our sense. In particular, the cluster head is still a bottleneck for both communication and computation, although not as severe as in a fully centralized scheme.

Diffusion method: Instead of doing full average consensus, the diffusion technique (e.g., Cattivelli & Sayed, 2010; Kanna et al., 2015; Song, Yu et al., 2014) approximates a global average with an average among the neighbouring nodes. This makes the algorithm fully distributed, but the estimation performance becomes sub-optimal.

Nearest neighbour interpolation-based method: In Martinez (2010), an asynchronous Kalman filter-like algorithm is proposed for distributed field estimation by combining a nearest neighbour interpolation method and Laplacian matrix-based average consensus algorithm. More specifically, each sensor produces a local estimate of a given field by fusing together its own measurement and information from neighbouring nodes via nearest neighbour interpolation, then these local estimates are fused together using a Laplacian matrix-based average consensus algorithm. Again, the required number of iterations for a given accuracy level cannot be determined *a priori*. In an example in Martinez (2010), for a network of 75 nodes, 25 iterations are used, which appears high. It is not clear how this scales up for a large network.

Kriging interpolation method: In Cortés (2009), a distributed field estimation algorithm is given based on the so-called Kriging interpolation technique. The algorithm employs the well-known Jacobi over-relaxation method and a dynamic average consensus method, both requiring an infinite number of iterations in theory. It is not clear how to choose the right number of iterations in practice, especially for large networks.

In summary, the cluster-based method gives the same estimation result for our example, but it requires the selection of the cluster head and centralized processing in the cluster, hence it is not a fully distributed method in our sense. The Kriging interpolation method gives a similar simulation result to the proposed

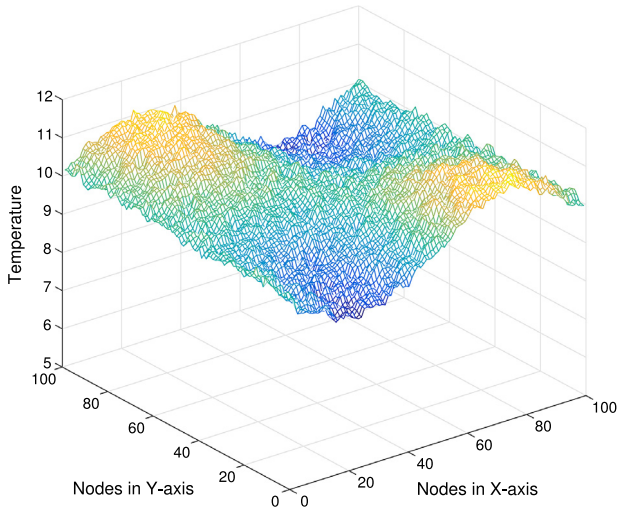

Fig. 7. Estimated temperature distribution using diffusion method.

method, when the iteration number is sufficiently large, but this is at the cost of more computation. The diffusion method and nearest neighbour interpolation-based method give compatible simulation results, which are shown in Fig. 7. We can see that they are inferior to the proposed method due to the fact that only the information from the direct neighbours is used for each node in each sampling point.

7. Conclusions

In this paper, we have proposed a new distributed algorithm for Kalman filtering for sensor networks. The algorithm runs very efficiently on acyclic network graphs, with very low complexities for each sensor node. It is also applicable to cyclic network graphs by running a distributed loop removal algorithm on the graph first, which is also a very efficient with very low complexities. The proposed algorithm provides the same estimation quality as a central Kalman filter and enjoys many extraordinary features including robustness against transmission adversaries, asynchronous implementability, and broadcast mode operation. The technical core of the algorithm is a distributed algorithm for maximum likelihood estimation, including weighted least-squares estimation as a special case. The algorithm is most suitable for detection, estimation and tracking of dynamic targets. With simple modifications, the algorithm is also suitable for an important application domain called distributed field estimation requiring low-complexity distributed solutions. We expect that this algorithm will find wide applications in large-scale sensor networks. Our simulation examples have demonstrated this potential. Future research topics include handling correlated measurement noises among sensors, transmission failures between sensors and quantization errors in the transmissions.

References

Anderson, B. D. O., & Moore, J. B. (1979). *Optimal filtering*. NJ: Prentice-Hall Englewood Cliffs.

Carli, R., Chiuso, A., Schenato, L., & Zampieri, S. (2008). Distributed Kalman filtering based on consensus strategies. *IEEE Journal on Selected Areas in Communications*, *26*(4), 622–633.

Cattivelli, F. S., & Sayed, A. H. (2010). Diffusion strategies for distributed Kalman filtering and smoothing. *IEEE Transactions on Automatic Control*, *55*(9), 2069–2084.

Chen, B., Zhang, W.-A., & Yu, L. (2014a). Distributed finite-horizon fusion Kalman filtering for bandwidth and energy constrained wireless sensor networks. *IEEE Transactions on Signal Processing*, *62*(4), 797–812.

Chen, B., Zhang, W.-A., & Yu, L. (2014b). Distributed fusion estimation with missing measurements, random transmission delays and packet dropouts. *IEEE Transactions on Automatic Control*, *59*(7), 1961–1967.

Chen, B., Zhang, W.-A., Yu, L., Hu, G., & Song, H. (2015). Distributed fusion estimation with communication bandwidth constraints. *IEEE Transactions on Automatic Control*, *60*(5), 1398–1403.

Cortés, J. (2009). Distributed Kriged Kalman filter for spatial estimation. *IEEE Transactions on Automatic Control*, *54*(12), 2816–2827.

Das, S., & Moura, J. M. F. (2015). Distributed Kalman filtering with dynamic observations consensus. *IEEE Transactions on Signal Processing*, *63*(17), 4458–4473.

Durrant-Whyte, H. F., & Rao, B. S. (1991). Fully decentralised algorithm for multisensor Kalman filtering. *IEE Proceedings D*, *138*(5), 413–420.

Goodrich, M. T., & Tamassia, R. (2001). *Algorithm design: Foundations, analysis, and internet examples*. Wiley.

Hashemipour, H. R., Sumit, R., & Laub, A. J. (1988). Decentralized structure for parallel Kalman filtering. *IEEE Transactions on Automatic Control*, *31*(1), 88–94.

Hu, J., Xie, L., & Zhang, C. (2012). Diffusion Kalman filtering based on covariance intersection. *IEEE Transactions on Signal Processing*, *60*(2), 891–902.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions ASME-Journal of Basic Engineering*, *82*(D), 35–45.

Kanna, S., Dini, D. H., Xia, Y., Hui, S. Y., & Mandic, D. P. (2015). Distributed widely linear Kalman filtering for frequency estimation in power networks. *IEEE Transactions on Signal and Information Processing over Networks*, *1*(1), 45–57.

Kar, S., & Moura, J. M. F. (2011). Gossip and distributed Kalman filtering: weak consensus under weak detectability. *IEEE Transactions on Signal Processing*, *59*(4), 1766–1784.

Khan, U. A., & Moura, J. M. F. (2008). Distributing the Kalman filter for large-scale systems. *IEEE Transactions on Signal Processing*, *56*(10), 4919–4935.

Li, D., Kar, S., Alsaadi, F. E., Dobaie, A. M., & Cui, S. (2015). Distributed Kalman filtering with quantized sensing state. *IEEE Transactions on Signal Processing*, *63*(19), 5180–5193.

Li, D., Kar, S., Moura, J. M. F., Poor, H. V., & Cui, S. (2015). Distributed Kalman filtering over massive data sets: analysis through large deviations of random Riccati equations. *IEEE Transactions on Information Theory*, *61*(3), 1351–1372.

Luo, B., & Wu, Y. C. (2013). Distributed clock parameters tracking in wireless sensor network. *IEEE Transactions on Wireless Communications*, *12*(12), 6464–6475.

Mahmoud, M. S., & Khalid, H. M. (2013). Distributed Kalman filtering: a bibliographic review. *IET Control Theory & Applications*, *7*(4), 483–501.

Marelli, D. E., & Fu, M. (2015). Distributed weighted least-squares estimation with fast convergence for large-scale systems. *Automatica*, *51*, 27–39.

Martinez, S. (2010). Distributed interpolation schemes for field estimation by mobile sensor networks. *IEEE Transactions on Control Systems Technology*, *18*(2), 491–500.

Medeiros, H., Park, J., & Kak, A. C. (2008). Distributed object tracking using a cluster-based Kalman filter in wireless camera networks. *IEEE Journal of Selected Topics in Signal Processing*, *2*(4), 448–463.

Nejad, B. M., Attia, S. A., & Raisch, J. (2009). Max-consensus in a max-plus algebraic setting: the case of fixed communication topologies. In *XXII International symposium on information, communication and automation technologies*.

Regazzoni, C. S. (1994). Distributed extended Kalman filtering network for estimation and tracking of multiple objects. *Electronics Letters*, *30*(15), 1202–1203.

Riberio, A., & Giannakis, G. B. (2006). Bandwidth-constrained distributed estimation for wireless sensor networks-Part I: Gaussian case. *IEEE Transactions on Signal Processing*, *54*(3), 1131–1143.

Riberio, A., Giannakis, G. B., & Roumeliotis, S. I. (2006). SOI-KF: Distributed Kalman filtering with low-cost communications using the sign of innovations. *IEEE Transactions on Signal Processing*, *54*(12), 4782–4795.

Roshany-Yamchi, S., Cychowski, M., Negenborn, R. R., De Schutter, B., Delaney, K., & Connell, J. (2013). Kalman filter-based distributed predictive control of large-scale multi-rate systems: application to power networks. *IEEE Transactions on Control Systems Technology*, *21*(1), 27–39.

Shen, X., Song, E., Zhu, Y., & Luo, Y. (2009). Globally optimal distributed Kalman fusion with local out-of-sequence-measurement updates. *IEEE Transactions on Automatic Control*, *54*(8), 1928–1934.

Skiena, S. (2008). *The algorithm design manual*. Springer.

Song, E., Xu, J., & Zhu, Y. (2014). Optimal distributed Kalman filtering fusion with singular covariances of filtering errors and measurement noises. *IEEE Transactions on Automatic Control*, *59*(5), 1271–1282.

Song, H., Yu, L., & Zhang, W.-A. (2014). Distributed consensus-based Kalman filtering in sensor networks with quantised communications and random sensor failures. *IET Signal Processing*, *8*(2), 107–118.

Sun, Y., Fu, M., Wang, B., & Zhang, H. (2015). A distributed MAP approach to dynamic state estimation with applications in power networks. In *European control conference*, July.

Tai, X., Lin, Z., Fu, M., & Sun, Y. (2013). A new distributed state estimation technique for power networks. In *American control conference*, Washington D.C., June.

Talarico, S., Schmid, N. A., Alkhweldi, M., & Valenti, M. C. (2014). Distributed estimation of a parametric field: algorithms and performance analysis. *IEEE Transactions on Signal Processing*, *62*(5), 1041–1053.

Vadigepalli, R., & Doyle III, F. J. (2003). A distributed state estimation and control algorithm for plantwide processes. *IEEE Transactions on Control Systems Technology*, *11*(1), 119–127.

Wang, S., Fang, H., & Liu, X. (2015). Distributed state estimation for stochastic nonlinear systems with random delays and packet dropouts. *IET Control Theory & Applications*, *9*(18), 2657–2665.

Wang, Y., Ishwar, P., & Saligrama, V. (2008). One-bit distributed sensing and coding for field estimation in sensor networks. *IEEE Transactions on Signal Processing*, *56*(9), 4433–4445.

Xiao, L., & Boyd, S. (2004). Fast linear iterations for distributed averaging. *Systems & Control Letters*, *53*, 65–78.

Xie, K., Cai, Q., Zhang, Z., & Fu, M. (2018). A fast convergent distributed algorithm for weighted average consensus. *IEEE Transactions on Automatic Control* submitted for publication.

Xu, J., Song, E., Luo, Y., & Zhu, Y. (2012). Optimal distributed Kalman filtering fusion algorithm without invertibility of estimation error and sensor noise covariances. *IEEE Signal Processing Letters*, *19*(1), 55–58.

Zhao, T., & Nehorai, A. (2007). Information-driven distributed maximum likelihood estimation based on Gauss-Newton method in wireless sensor networks. *IEEE Transactions on Signal Processing*, *55*(9), 4669–4682.

Zhou, T. (2013). Coordinated one-step optimal distributed state prediction for a networked dynamical system. *IEEE Transactions on Automatic Control*, *58*(11), 2756–2771.

Zhou, Z., Fang, H., & Hong, Y. (2013). Distributed estimation for moving target based on state-consensus strategy. *IEEE Transactions on Automatic Control*, *58*(8), 2096–2101.

Zongze Wu received his B.S. degree in material forming and control, M.S. degree in control science and engineering, and Ph.D. degree in pattern reorganization and intelligence system, all from Xi'an Jiaotong University, Xi'an, China, in 1999, 2002, and 2005, respectively. He is currently a professor of the School of Automation, Guangdong University of Technology, Guangzhou, China. His research interests include Automation Control, Signal Processing, Big Data and Internet of Things. He has served as the Under-Secretary-General for Internet of Things and Information Technology Innovation Alliance in Guangdong Province, China. Dr. Wu was the recipient of the Microsoft Fellowship Award of the MSRA in 2003. He won the Technological Award first prize of Guangdong Province three times in 2008, 2013 and 2014, respectively. He got second prize of Ministry of Education technological innovation twice, in 2012 and 2013.

Minyue Fu received his Bachelor's Degree in Electrical Engineering from the University of Science and Technology of China, Hefei, China, in 1982, and M.S. and Ph.D. degrees in Electrical Engineering from the University of Wisconsin-Madison in 1983 and 1987, respectively. From 1983 to 1987, he held a teaching assistantship and a research assistantship at the University of Wisconsin-Madison. He worked as a Computer Engineering Consultant at Nicolet Instruments, Inc., Madison, Wisconsin, during 1987. From 1987 to 1989, he served as an Assistant Professor in the Department of Electrical and Computer Engineering, Wayne State University, Detroit, Michigan. He joined the Department of Electrical and Computer Engineering, the University of Newcastle, Australia, in 1989. Currently, he is a Chair Professor in Electrical Engineering and Head of School of Electrical Engineering and Computer Science. In addition, he was a Visiting Associate Professor at University of Iowa in 1995–1996, and a Senior Fellow/Visiting Professor at Nanyang Technological University, Singapore, 2002. He has held a Qianren Professorship at Zhejiang University and Guangdong University of Technology, China. He is a Fellow of IEEE. His main research interests include control systems, signal processing and communications. He has been an Associate Editor for the IEEE Transactions on Automatic Control, Automatica and Journal of Optimization and Engineering.

Yong Xu was born in Zhejiang Province, China, in 1983. He received the B.S. degree in information engineering from Nanchang Hangkong University, Nanchang, China, in 2007, the M.S. degree in control science and engineering from Hangzhou Dianzi University, Hangzhou, China, in 2010, and the Ph.D. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2014. He was a visiting internship student with the department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, from June 2013 to November 2013. He was honored Pearl River Young Scholars Program of Guangdong Province in 2017. Now he is an associate professor with School of Automation, at Guangdong University of Technology, Guangzhou, China.

His research interests include networked control systems, state estimation, and positive systems.

Renquan Lu received his Ph.D. degree in Control Science and Engineering from Zhejiang University, Hangzhou, China, in 2004. He was supported by the National Science Fund for Distinguished Young Scientists of China in 2014, honored as the Distinguished Professor of Pearl River Scholars Program of Guangdong Province, the Distinguished Professor of Yangtze River Scholars Program by the Ministry of Education of China in 2015 and 2017, respectively. Currently, he is a professor of the School of Automation at Guangdong University of Technology, Guangzhou, China. His research interests include complex systems, networked control systems, and nonlinear systems.