Distributed Kalman Filtering via Node Selection in Heterogeneous Sensor Networks

Donato Di Paola[a], Antonio Petitti[a,b], Alessandro Rizzo[b,c] *

[a] *Institute of Intelligent Systems for Automation (ISSIA), National Research Council (CNR), 70126 Bari, Italy*;

[b] *Dipartimento di Ingegneria Elettrica e dell' Informazione (DEI), Politecnico di Bari, 70126 Bari, Italy*

[c] *Department of Mechanical and Aerospace Engineering, New York University Polytechnic School of Engineering, Six MetroTech Center, Brooklyn, NY 11021, USA*

(*Received 16 May 2013; accepted 23 September 2013*)

In this paper, we propose a strategy for distributed Kalman filtering over sensor networks, based on node selection, rather than on sensor fusion. The presented approach is particularly suitable when sensors with limited sensing capability are considered. In this case, strategies based on sensor fusion may exhibit poor results, as several unreliable measurements may be included in the fusion process. On the other hand, our approach implements a distributed strategy able to select only the node with the most accurate estimate and to propagate it through the whole network in finite time. The algorithm is based on the definition of a metric of the estimate accuracy, and on the application of an agreement protocol based on max-consensus. We prove the convergence, in finite time, of all the local estimates to the most accurate one at each discrete iteration, as well as the equivalence with a centralised Kalman filter with multiple measurements, evolving according to a state-dependent switching dynamics. An application of the algorithm to the problem of distributed target tracking over a network of heterogeneous range-bearing sensors is shown. Simulation results and a comparison with two distributed Kalman filtering strategies based on sensor fusion confirm the suitability of the approach.

Keywords: Distributed Kalman Filtering, Distributed Estimation, Node Selection, Sensor Networks, Target Tracking

1. Introduction

In the last decades, sensor networks have received significant attention by many researchers in different fields of engineering, such as robotics, control theory, and image processing (Zhao and Guibas (2004)). As a general principle, sensor networks perform estimates of the state of dynamical processes through computation and communication among the network nodes. Recent technological innovations have made possible to deploy a large number of inexpensive and low-power sensors to cover wide areas (Dargie and Poellabauer (2010)), making a wide range of applications possible and affordable, such as habitat monitoring, animal tracking, environment observation, forecasting, surveillance, and domotics.

The communication scheme of a sensor network determines whether the estimation algorithm is centralised, distributed or hierarchical. Centralised algorithms make use of a fusion centre, able to receive data from each node and to compute a global estimate of the system state (Ghaffarkhah and Mostofi (2009), Taj and Cavallaro (2009)). The presence of a single fusion centre simplifies the process of computing a unique and optimised estimate. On the other hand, this may lead to high data transfer rates and to lack of scalability, energy efficiency, and fault tolerance, because of the presence of a single point of failure. These limitations can be overcome through *distributed* approaches (Mahmoud and Khalid (2013)). Steps towards the realisation of effective distributed approaches have been at first made by considering all-to-all communication

schemes over the sensor network, which yield the elimination of the fusion centre, but may introduce a heavy communication overhead (Rao et al. (1993)). Further approaches consider local connection schemes, by making use of local fusion centres which communicate only with subsets of nodes (Oruc et al. (2009), Wang and Wu (2008)). In this case, *hierarchical* or *hybrid* architectures come into action. The transition to fully distributed estimation algorithms requires the definition of suitable agreement procedures, able to lead all the nodes towards the production of a common estimate without the support of centralizing units, and possibly with limited communication and computation burdens. To this aim, consensus algorithms (Cao and Ren (2010), Ren and Beard (2007)) have been widely adopted. Remarkable applications of consensus algorithms have been carried out in different contexts, such as task coordination (Olfati-Saber et al. (2007), Ren et al. (2005, 2007)), task assignment (Di Paola et al. (2011)), formation control (Fax and Murray (2004)), flocking (Buscarino et al. (2006, 2007), La and Sheng (2012), Olfati-Saber (2006)), rendez-vous (Martinez et al. (2005)), and distributed estimation (Millán et al. (2013), Yang et al. (2011), Yang and Shi (2012), Yu et al. (2011)).

The transition from centralised to distributed algorithms has touched nearly all the fields of estimation over networks, including the well-known Kalman filter (Kalman (1960)). The Centralised Kalman Filter with multiple measurements (CKF) is, in fact, a centralised estimation strategy which can be implemented over sensor networks (Mitchell (2007)). Further, much effort has been devoted to the implementation of distributed estimation schemes based on Kalman filtering, such as the Kalman-Consensus Filter (KCF) (Olfati-Saber (2007)), or the Diffusion Kalman Filter (DKF) (Cattivelli and Sayed (2010)). Nevertheless, these approaches do not guarantee the convergence of all the individual estimates to a common value, and require a further phase of sensor fusion, even though this is performed on a subset of the network nodes, with the aim of containing the communication and computational burden. It is well known that special care must be taken in sensor fusion, when sensors with *limited sensing capability* are considered, as the phenomenon of *catastrophic fusion* may occur (Mitchell (2007)). Limited sensing capability is a realistic scenario in applications, such as, for example, indoor surveillance and target tracking. There, sensors have usually a limited sensing range and, at a given time instant, it may happen that the majority of sensors does not even possess a measurement. With this in mind, one can easily figure out the effect of involving many unreliable sensors in the fusion operation. Therefore, it may be the case that particular operational conditions, such as that of limited sensing capability, call for a different strategy in the computation of a unique estimate from the network data, giving up sensor fusion to go in favour of *node selection* (Yang and Shi (2012)). This means that only some, if not a single network node is selected as the holder of the best estimate, which is propagated to all the network nodes.

In this paper, we introduce a fully distributed approach to the realisation of an algorithm for Distributed Kalman filtering with Node Selection (DKNS) for heterogeneous sensor networks, where nodes have limited sensing capability. Considering heterogeneous nodes has important consequences in the algorithm design, since the accuracy of the measurements performed by each sensor may change over space and time, thus involving different uncertainties on estimates to be accounted for during the agreement process over the network.

Beyond being fully distributed, another viable property of DKNS is that it guarantees that at each iteration, every network node owns the same estimate of the process state. Node selection is achieved using a strategy based on the *max-consensus* protocol (Nejad et al. (2009)), performed on a measurement accuracy metric called *perception confidence value,* strictly related to the Fisher information (Grocholsky et al. (2003), Martinez and Bullo (2006)). The approach is tested on a distributed target tracking application (Giannini et al. (2013), Petitti et al. (2011)), where a realistic range-bearing sensor model is considered. Simulation results are satisfactory, and a comparison with two distributed Kalman filtering strategies, namely those in (Cattivelli and Sayed (2010), Olfati-Saber (2007)), confirms that our approach is more effective when substantial limitations to the sensing capability are present.

The paper is organised as follows. Section 2 explains in full detail the proposed algorithm. In Section 3, the performance of the proposed algorithm is assessed via numerical simulations on the distributed target tracking application. Finally, conclusions and future directions are drawn.

2. Distributed Kalman Filtering via Node Selection

This section presents the Distributed Kalman filtering with Node Selection (DKNS) algorithm, starting from the node and network model, to the description of its working principle, ending with some discussion on DKNS properties.

2.1. *Sensor Network and Node Model*

We consider a sensor network modeled by an undirected graph $\mathcal{G} = \{\mathcal{I}, \mathcal{E}\}$, where $\mathcal{I} \stackrel{\triangle}{=} \{1, \ldots, n\}$ is the node set, and $\mathcal{E} \subseteq \mathcal{I} \times \mathcal{I}$ is the set of edges (links), representing a point-to-point communication network. The neighbour set of node $i \in \mathcal{I}$ is defined as $\mathcal{N}_i = \{j \in \mathcal{I} \mid (i,j) \in \mathcal{E}\}$. Each network node $i \in \mathcal{I}$ is a sensor able to estimate the state $\mathbf{x}(k) \in \mathbb{R}^m$ of a time-discrete process through a Kalman filter, and to exchange information only with its one-hop neighbours. In accordance with the classic Kalman filter theory, we assume that the process state $\mathbf{x}(k)$ evolves according to a linear model, as

$$\mathbf{x}(k) = \mathbf{A}(k)\mathbf{x}(k-1) + \boldsymbol{\omega}(k), \tag{1}$$

where $\mathbf{A}(k)$ is the state transition matrix and $\boldsymbol{\omega}(k)$ is the process noise, which is assumed to be drawn from a zero mean multivariate normal distribution with covariance matrix $\mathbf{Q}(k)$, that is, $\boldsymbol{\omega}(k) \sim N(0, \mathbf{Q}(k))$.

Moreover, we assume that measurements $\mathbf{z}_i(k)$ of the state $\mathbf{x}(k)$ are performed by the generic i-th node as:

$$\mathbf{z}_i(k) = \mathbf{H}_i(k)\mathbf{x}(k) + \boldsymbol{\upsilon}_i(k), \tag{2}$$

where $\mathbf{H}_i(k)$ is the measurement output matrix of the i-th node and $\boldsymbol{\upsilon}_i(k)$ is the related observation noise, which is assumed to be zero mean Gaussian white, with covariance matrix $\mathbf{R}_i(k)$, that is, $\boldsymbol{\upsilon}_i(k) \sim N(0, \mathbf{R}_i(k))$. The noise vectors at each time step are assumed to be mutually independent.

The generic node $i \in \mathcal{I}$ is represented by the tuple:

$$\langle \hat{\mathbf{x}}_i(k),\, \mathcal{X}_i(k),\, \gamma_i(k),\, \overline{\mathbf{x}}_i(k) \rangle. \tag{3}$$

In (3), $\hat{\mathbf{x}}_i(k)$ is the local state estimate produced by the sensor i at the discrete time k. To model the limited sensing capability of each sensor, the *sensing set* $\mathcal{X}_i(k) \subset \mathbb{R}^m$ is defined, so that node i is able to perform a measurement of the state $\mathbf{x}(k)$ if and only if $\mathbf{x}(k) \in \mathcal{X}_i(k)$. Heterogeneity of sensors is assumed, by defining individual sensing sets. We define the set of the *sensing* nodes $\mathcal{S}(k) \stackrel{\triangle}{=} \{i \in \mathcal{I} \,|\, \mathbf{x}(k) \in \mathcal{X}_i(k)\}$. The set difference $\mathcal{P}(k) = \mathcal{I} \setminus \mathcal{S}(k)$ is defined as the set of *predicting* nodes. The quantity $\gamma_i(k)$ is dubbed as *perception confidence value* and quantifies the estimate accuracy of node i. To work effectively, the perception confidence value must have the property of being larger as long as the estimate is more accurate. More formally, considering two state estimates $\hat{\mathbf{x}}_i(k)$ and $\hat{\mathbf{x}}_j(k)$, where i and $j \in \mathcal{I}$; if $E[||\hat{\mathbf{x}}_i(k) - \mathbf{x}(k)||^2] < E[||\hat{\mathbf{x}}_j(k) - \mathbf{x}(k)||^2]$, then $\gamma_i(k) > \gamma_j(k)$ must hold (where $E[\cdot]$ indicates statistical expectation). Finally, $\overline{\mathbf{x}}_i(k)$ is the value of the state estimate obtained after the procedure of node selection.

Algorithm 1 Estimation phase for node i at iteration k

Input: $\overline{\mathbf{x}}_i(k-1)$, $\overline{\mathbf{P}}_i(k-1)$
Output: $\gamma_i(k)$, $\hat{\mathbf{x}}_i(k)$, $\mathbf{\Psi}_i(k)$

1: $\hat{\mathbf{x}}_i^-(k) = \mathbf{A}(k)\overline{\mathbf{x}}_i(k-1)$
2: $\mathbf{P}_i^-(k) = \mathbf{A}(k)\overline{\mathbf{P}}_i(k-1)\mathbf{A}^T(k) + \mathbf{Q}(k)$
3: **if** $i \in \mathcal{S}(k)$ **then**
4: $\quad[\mathbf{P}_i(k)]^{-1} = [\mathbf{P}_i^-(k)]^{-1} + \mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{H}_i(k)$
5: $\quad\hat{\mathbf{x}}_i(k) = \mathbf{P}_i(k)\{[\mathbf{P}_i^-(k)]^{-1}\hat{\mathbf{x}}_i^-(k) + \mathbf{H}_i^T(k)\mathbf{R}_i^{-1}(k)\mathbf{z}_i(k)\}$
6: $\quad\mathbf{\Psi}_i(k) = \mathbf{P}_i(k)$
7: **else**
8: $\quad\hat{\mathbf{x}}_i(k) = \hat{\mathbf{x}}_i^-(k)$
9: $\quad\mathbf{\Psi}_i(k) = \mathbf{P}_i^-(k)$
10: **end if**
11: $\gamma_i(k) = 1/\mathrm{Trace}\,[\mathbf{\Psi}_i(k)]$

Each iteration of DKNS consists of two phases: *estimation* and *node selection.* In the former phase, each node performs an individual estimate through a local Kalman filter. In the latter, all the nodes agree on the best available estimate in a distributed fashion. We now describe the two phases in detail at iteration k, therefore assuming that each node owns the information of the tuple (3) at iteration $k-1$.

2.2. *Phase 1: Estimation Phase*

The estimation phase is schematised in Algorithm **1**. Each node locally runs a Kalman filter. Specifically, the local Kalman filter starts, at each iteration, from a given state estimate $\overline{\mathbf{x}}_i(k-1)$ and covariance matrix $\overline{\mathbf{P}}_i(k-1)$, which are the best ones available over the network, obtained and propagated during iteration $k-1$. If at iteration k, node i is a *sensing* node, *i.e.,* $i \in \mathcal{S}(k)$, a full Kalman filter iteration is run (lines 1–2 and 4–6 of Algorithm **1**); on the contrary, in the case of a *predicting* one, *i.e.,* $i \in \mathcal{P}(k)$, only the prediction part of the Kalman filter is run (lines 1–2 and 8–9). Then, each node ends the estimation phase by computing the perception confidence value $\gamma_i(k)$ as:

$$\gamma_i(k) \triangleq \begin{cases} \frac{1}{\mathrm{Trace}[\mathbf{P}_i(k)]} & \text{if } i \in \mathcal{S}(k) \\ \frac{1}{\mathrm{Trace}[\mathbf{P}_i^-(k)]} & \text{if } i \in \mathcal{P}(k) \end{cases}, \tag{4}$$

that is, if node i is a sensing one, the perception confidence value $\gamma_i(k)$ is computed on the basis of the *a posteriori* error covariance matrix, otherwise, the operation refers to the *a priori* error covariance matrix. It is well known that minimizing the trace of the error covariance matrix is equivalent to minimize the expected value of the square of the magnitude of the error vector (Siouris (1979)); therefore it is clear that $\gamma_i(k)$ grows with the accuracy of the estimation performed by the node i at time k, thus, computing the perception confidence value as in Eq. (4) fulfils the definition given in Section 2.1.

Eventually, at the end of Phase 1, each node i stores the perception confidence value $\gamma_i(k)$, the local state estimate $\hat{\mathbf{x}}_i(k)$, and the local error covariance matrix $\mathbf{\Psi}_i(k)$ (output line of Algorithm **1**).

Algorithm 2 Node Selection phase for node i at iteration k

Input: $\gamma_i(k)$, $\hat{\mathbf{x}}_i(k)$, $\mathbf{\Psi}_i(k)$
Output: $\overline{\mathbf{x}}_i(k)$, $\overline{\mathbf{P}}_i(k)$

1: $\zeta_i(0) = \gamma_i(k)$
2: $\boldsymbol{\chi}_i(0) = \hat{\mathbf{x}}_i(k)$
3: $\mathbf{\Pi}_i(0) = \mathbf{\Psi}_i(k)$
4: **for** $t = 1$ to D **do**
5: $\quad$ SEND$(\zeta_i(t-1), \boldsymbol{\chi}_i(t-1), \mathbf{\Pi}_i(t-1))$
6: $\quad$ RECEIVE$(\zeta_j(t-1), \boldsymbol{\chi}_j(t-1), \mathbf{\Pi}_j(t-1)), \quad \forall j \in \mathcal{N}_i$
7: $\quad \zeta_i(t) = \max_{j \in \mathcal{N}_i \cup i} \{\zeta_j(t-1)\}$
8: $\quad \mu_i(t) = \text{argmax}_{j \in \mathcal{N}_i \cup i} \{\zeta_j(t-1)\}$
9: $\quad \boldsymbol{\chi}_i(t) = \boldsymbol{\chi}_{\mu_i}(t-1)$
10: $\quad \mathbf{\Pi}_i(t) = \mathbf{\Pi}_{\mu_i}(t-1)$
11: **end for**
12: $\overline{\mathbf{x}}_i(k) = \boldsymbol{\chi}_i(D)$
13: $\overline{\mathbf{P}}_i(k) = \mathbf{\Pi}_i(D)$

2.3. *Phase 2: Node Selection Phase*

The node selection phase, schematised in Algorithm **2**, is run to select, in a totally distributed fashion, the node with the highest perception confidence value, and to propagate its estimate and the related error covariance matrix over the network in finite time. To achieve this, a max-consensus based strategy (Nejad et al. (2009)) is run in lines 4–11 of Algorithm **2**. Please note that in this phase of the algorithm, a new discrete time index t is defined, making the max-consensus protocol run on a different sampling time (typically, shorter than the one associated with the time-discrete index k, which is the actual sampling time of DKNS).

Algorithm **2** works as follows: node i initially sets its internal variables to values obtained from Phase 1, *i.e.,* the perception confidence value, the state estimate, and the covariance matrix (line 1–3 of Algorithm **2**). Then, a protocol based on max-consensus lets the internal variable $\zeta_i(\cdot)$ to converge to the maximum of all perception confidence values over the network (line 7). Correspondingly, line 8 selects the node which performed the estimate of the process state with the best accuracy, whereas lines 9 and 10 select the corresponding best estimate and error covariance matrix. It can be proved (Nejad et al. (2009)) that the node selection loop (lines 4–11) converges in at most D steps, where D is the diameter of the graph $\mathcal{G}$ (Bondy and Murty (2008)). Convergence issues will be dealt with later in this section. For each iteration t of the node selection loop, node i shares the variables $\zeta_i(t-1), \boldsymbol{\chi}_i(t-1), \mathbf{\Pi}_i(t-1)$ with its neighbours through a send/receive procedure (lines 5–6). After line 7, which implements the dynamics of the max-consensus protocol, $\mu_i(D)$ will take the index of the node which performed the best estimate (line 8), $\boldsymbol{\chi}_i(D)$, and $\mathbf{\Pi}_i(D)$ will take the corresponding estimate and covariance estimate matrices, respectively (line 9–10). At the end of Phase 2, an agreement both on the best estimate of the process state and on the related *a posteriori* covariance matrix is achieved. Each node will store these values in its variables $\overline{\mathbf{x}}_i(k)$ and $\overline{\mathbf{P}}_i(k)$ (lines 12–13), respectively. These two variables will be fed back to the next iteration $(k+1)$ of Phase 1, in order to let the Kalman filter of each node start from the best available estimate, and therefore to improve the individual prediction performance.

Remark 1 : Line 7 of Algorithm **2** requires a tie-break rule in the case that the maximum value for $\zeta_j(t-1)$, $j \in \mathcal{N}_i$, is allocated in more than one of the neighboring nodes. The tie-break rule must guarantee convergence and has to be performed in a distributed way. Examples of this kind of tie-break rules are: choosing an index at random among those corresponding to the maximum, or performing, on lines 9 and 10, an average of $\boldsymbol{\chi}_{\mu_i}$ and $\mathbf{\Pi}_{\mu_i}$ along the involved indices

μ_i, rather than a variable assignment.

Remark 2 : It is important to note that yet predicting nodes can be selected as those possessing the best estimate of the process state. In the case of heterogeneous sensor networks, in fact, it is possible that a predicting node that makes a prediction starting from a very accurate past measurement performs better than a sensing node which is directly measuring with poor accuracy.

2.4. *Convergence Properties*

The DKNS algorithm makes each node converge in finite time to the estimate of the process state owned by the node with the highest perception confidence value, at each discrete time step k. This is made possible thanks to the convergence property of the max-consensus protocol (Nejad et al. (2009)), which are here briefly recalled before providing our main result on the DKNS convergence.

Theorem 2.1 : *Consider a network of n dynamical systems, connected over an undirected graph $\mathcal{G} = \{\mathcal{I}, \mathcal{E}\}$. Each dynamical system has a state variable $\varsigma_i(t) \in \mathbb{R}$, $i \in \mathcal{I}$, $t \in \mathbb{N}_0$. The discrete time max-consensus protocol is defined as*

$$\varsigma_i(t+1) = \max_{j \in \mathcal{N}_i \bigcup i} \varsigma_j(t). \tag{5}$$

Assume that each node of the network $\mathcal{G}$ runs the protocol in Eq.(5). If $\mathcal{G}$ is connected, then

$$\begin{aligned} \varsigma_i(t) &= \varsigma_j(t) \\ &= \max\left(\varsigma_1(0), \cdots, \varsigma_n(0)\right) \quad \forall i, j \in \mathcal{I}, \quad \forall t \geq D, \end{aligned}$$

where D is the graph diameter.

Proof : The proof is provided in Theorem 4.1 of (Nejad et al. (2009)). □

The following theorem proves the convergence of the DKNS.

Theorem 2.2 : *Given a network of n nodes modeled as an undirected connected graph $\mathcal{G} = \{\mathcal{I}, \mathcal{E}\}$, and assuming that each node runs the DKNS algorithm; then, at the end of each discrete iteration $k \in \mathbb{N}_0$, the following holds: $\overline{\mathbf{x}}_i(k) = \overline{\mathbf{x}}_j(k) = \overline{\mathbf{x}}_{j^\star(k)}(k) \quad \forall\, i, j \in \mathcal{I}$, and $j^\star(k) = \operatorname*{argmax}_{j \in \mathcal{I}} \gamma_j(k)$.*

Proof : The proof comes from the analysis of the steps of Algorithm **2**. Note that line 7 in Algorithm **2** is the update rule of the discrete time max-consensus protocol over the perception confidence values of each node of the network (assigned to variables $\zeta_i(t)$ in Algorithm **2**). Theorem 2.1 proves that this protocol converges to the maximum of the initial states in a finite number of steps, upper bounded by the graph diameter D, if and only if $\mathcal{G}$ is connected. Therefore, in D steps, it is guaranteed that

$$\zeta_i(D) = \zeta_j(D) = \max_{l \in \mathcal{I}} \zeta_l(0) = \max_{l \in \mathcal{I}} \gamma_l(k), \quad \forall\, i, j \in \mathcal{I}.$$

Correspondingly, in D steps, the assignment in line 8 allows each node to store the index of the node where the maximum of the perception confidence values is located, *i.e.*, $\mu_i(D) = j^\star(k) = \operatorname*{argmax}_{l \in \mathcal{I}} \gamma_l(k)$. Thus, $\boldsymbol{\chi}_i(D) = \boldsymbol{\chi}_{j^\star(k)}$ and $\boldsymbol{\Pi}_i(D) = \boldsymbol{\Pi}_{j^\star(k)}$, $\forall i \in \mathcal{I}$. Finally, the assignment in line 12 concludes the proof of the theorem. □

Remark 3 : A single iteration of DKNS consists of two cascaded phases, therefore its duration is the sum of the durations of the two phases. The estimation phase consists of a sequence of variable assignments, therefore it terminates in finite time, indicated as t_e. Taking into account the result of Theorem 2.2, the convergence of the node selection phase is guaranteed if the node selection loop (lines 4–11 in Algorithm **2**) is run for at least D steps. The graph diameter D can be easily computed, even in a totally distributed framework (Cardoso et al. (2009)). Real time applications of DKNS require that the node selection phase is completed before a new instance of the estimation phase is started. Thus, defining ε the sampling time associated to the single iteration of the DKNS algorithm (related to discrete time index k), and τ the sampling time associated to one iteration of the node selection loop (*i.e.,* associated to the discrete time index t) in Algorithm **2**, the inequality $t_e + D\tau < \varepsilon$, must hold. We also remark that the diameter of the graph D is related to the number of nodes n and to the graph topology. Thus, the network structure can concur to impose constraints to the choice of the sampling time ε.

2.5. *Analysis of the Node Selection Mechanism*

Here, the equivalence between the DKNS and a particular Centralised Kalman Filter (CKF) is shown.

In CKF with multiple measurements (Mitchell (2007)), a fusion centre collects the measurements, $\mathbf{z}_i(k)$, the measurement output matrices, $\mathbf{H}_i(k)$, and the observation covariance matrices, $\mathbf{R}_i(k)$, from all nodes i, $i = 1, \ldots, n$. Then, a Kalman filter is run at the fusion centre level (Anderson and Moore (1979)). In distributed approaches, no centralisation of the information is admitted. Hence, each node can communicate with a limited set of neighbours. Distributed Kalman filters have been proposed in literature to overcome the limitations of centralised approaches; yet retaining some aspects of hierarchical organisation (Cattivelli and Sayed (2010), Olfati-Saber (2007)). Distributed versions of CKF needs to locally approximate the covariance matrix

$$[\mathbf{P}(k)]^{-1} = [\mathbf{P}^{-}(k)]^{-1} + \sum_{j=1}^{n} \mathbf{H}_j^T(k)\mathbf{R}_j^{-1}(k)\mathbf{H}_j(k) \tag{6}$$

at the node level [1]. It is straightforward to note that in CKF all the sensors contribute to the update of the *a posteriori* covariance matrix. On the other hand, in distributed approaches each node approximates locally the summation in Eq. (6). For example, this is done through a local summation within the neighbourhood of each node, as

$$[\mathbf{P}_i(k)]^{-1} = [\mathbf{P}_i^{-}(k)]^{-1} + \sum_{j \in \mathcal{N}_i} \mathbf{H}_j^T(k)\mathbf{R}_j^{-1}(k)\mathbf{H}_j(k), \tag{7}$$

where $[\mathbf{P}_i(k)]^{-1}$ is the local estimate of the centralised *a posteriori* inverse covariance matrix (6) carried out by node i, and $\mathcal{N}_i$ is the set of the direct neighbours of node i.

Theorem 2.3 : *Given a network of n nodes modelled as an undirected connected graph $\mathcal{G} = \{\mathcal{I}, \mathcal{E}\}$, and assuming that each node has limited sensing capability, that is, the measurement set is given by $\mathbf{z}_j(k) = \mathbf{H}_j(k)\mathbf{x}(k) + \boldsymbol{\upsilon}_j(k)$, with $j \in \mathcal{S}(k)$; then, running DKNS is equivalent to run a Centralised Kalman Filter with multiple measurements and state-dependent switching dynamics,*

[1] Please note that the update of the covariance matrix is written by using the well known information form of the Kalman filter (Anderson and Moore (1979)).

whose evolution is dictated by:

$$\hat{\mathbf{x}}^-(k) = \mathbf{A}(k)\hat{\mathbf{x}}(k-1), \tag{8}$$

$$\mathbf{P}^-(k) = \mathbf{A}(k)\mathbf{P}(k-1)\mathbf{A}^T(k) + \mathbf{Q}(k), \tag{9}$$

$$[\mathbf{P}(k)]^{-1} = [\mathbf{P}^-(k)]^{-1} + \mathbb{1}_{\mathcal{X}_{j^\star(k)}}(\mathbf{x}(k))[\mathbf{H}^T_{j^\star(k)}(k)\mathbf{R}^{-1}_{j^\star(k)}(k)\mathbf{H}_{j^\star(k)}(k)], \tag{10}$$

$$\hat{\mathbf{x}}(k) = \mathbf{P}(k)\{[\mathbf{P}^-(k)]^{-1}\hat{\mathbf{x}}^-(k) + \mathbb{1}_{\mathcal{X}_{j^\star(k)}}(\mathbf{x}(k))[\mathbf{H}^T_{j^\star(k)}(k)\mathbf{R}^{-1}_{j^\star(k)}(k)\mathbf{z}_{j^\star(k)}(k)]\}; \tag{11}$$

where $j^\star(k) = \underset{j\in\mathcal{I}}{\operatorname{argmax}}\ \gamma_j(k)$,

$$\gamma_j(k) \triangleq \begin{cases} \frac{1}{\mathrm{Trace}[\mathbf{P}_j(k)]} & \text{if } \mathbf{x}(k) \in \mathcal{X}_j(k) \\ \frac{1}{\mathrm{Trace}[\mathbf{P}^-_j(k)]} & \text{if } \mathbf{x}(k) \notin \mathcal{X}_j(k) \end{cases},$$

$\mathbf{P}^-_j(k)$, $\mathbf{P}_j(k)$ are, respectively, the *a priori* and *a posteriori* error covariance matrices associated to the measurement $\mathbf{z}_j(k)$, $j \in \mathcal{S}(k)$, and $\mathbb{1}_{\mathcal{X}_i(k)}(\cdot)$ is the indicator function of the set $\mathcal{X}_i(k)$, $\mathbb{1}_{\mathcal{X}_i(k)} : \mathbb{R}^m \to \{0,1\}$, defined as

$$\mathbb{1}_{\mathcal{X}_i(k)}(\mathbf{x}(k)) \triangleq \begin{cases} 1 & \text{if } \mathbf{x}(k) \in \mathcal{X}_i(k) \\ 0 & \text{if } \mathbf{x}(k) \notin \mathcal{X}_i(k) \end{cases}.$$

Proof: The proof comes from the analysis of the steps of Algorithms **1** and **2**. Let us focus on iteration k. Thanks to Theorem 2.2, we already proved that, at the start of Algorithm **1**, all the nodes in the network own the state estimate and the covariance matrix of the node selected at the end of iteration $k-1$ of Algorithm **2**, that is, $\overline{\mathbf{x}}_i(k-1) = \hat{\mathbf{x}}_{j^\star(k-1)}(k-1)$, $\overline{\mathbf{P}}_i(k-1) = \mathbf{P}_{j^\star(k-1)}(k-1)$, $\forall i \in \mathcal{I}$. Lines 1 and 2 of Algorithm **1** are common to all the network nodes, thus all the *a priori* state estimations and *a priori* error covariance matrices are identical, for any node $i \in \mathcal{I}$, and are given by

$$\begin{cases} \hat{\mathbf{x}}^-_i(k) = \mathbf{A}(k)\overline{\mathbf{x}}_i(k-1) \\ \mathbf{P}^-_i(k) = \mathbf{A}(k)\overline{\mathbf{P}}_i(k-1)\mathbf{A}^T(k) + \mathbf{Q}(k) \end{cases}, \quad \forall i \in \mathcal{I}. \tag{12}$$

The computation of the *a posteriori* differs, on the other hand, between *sensing* and *predicting* nodes. Namely, for sensing nodes,

$$\begin{cases} [\mathbf{P}_i(k)]^{-1} = [\mathbf{P}^-_i(k)]^{-1} + \mathbf{H}^T_i(k)\mathbf{R}^{-1}_i(k)\mathbf{H}_i(k) \\ \hat{\mathbf{x}}_i(k) = \mathbf{P}_i(k)\{[\mathbf{P}^-_i(k)]^{-1}\hat{\mathbf{x}}^-_i(k) + \mathbf{H}^T_i(k)\mathbf{R}^{-1}_i(k)\mathbf{z}_i(k)\} \end{cases}, \quad \forall i \in \mathcal{S}(k),$$

is computed, while, the following holds for predicting nodes

$$\begin{cases} \mathbf{P}_i(k) = \mathbf{P}^-_i(k) \\ \hat{\mathbf{x}}_i(k) = \hat{\mathbf{x}}^-_i(k) \end{cases}, \quad \forall i \in \mathcal{P}(k).$$

Each node possesses its own perception confidence value, so that, $\gamma_i(k)$, $\forall i \in \mathcal{I}$, is available to Algorithm **2**. Theorem 2.2 proves that, in finite time, all the nodes store in $\overline{\mathbf{x}}_i(k)$ and in $\overline{\mathbf{P}}_i(k)$, in finite time, the state estimation and its corresponding error covariance matrix of a specific node, that is, node $j^\star(k) = \underset{j\in\mathcal{I}}{\operatorname{argmax}}\ \gamma_j(k)$. It can be easily verified that, if $j^\star(k)$ is a sensing node, that is, $\mathbf{x}(k) \in \mathcal{X}_{j^\star(k)}$,

$$\begin{cases} [\overline{\mathbf{P}}_i(k)]^{-1} = [\mathbf{P}_i^-(k)]^{-1} + [\mathbf{H}_{j^\star(k)}^T(k)\mathbf{R}_{j^\star(k)}^{-1}(k)\mathbf{H}_{j^\star(k)}(k)] \\ \overline{\mathbf{x}}_i(k) = \overline{\mathbf{P}}_i(k)\{[\mathbf{P}_i^-(k)]^{-1}\mathbf{x}_i^-(k) + \mathbf{H}_{j^\star(k)}^T(k)\mathbf{R}_{j^\star(k)}^{-1}(k)\mathbf{z}_{j^\star(k)}(k)\} \end{cases}, \quad \forall i \in \mathcal{I}, \tag{13}$$

is obtained. On the other hand, if $j^\star(k)$ is a predicting node, that is, $\mathbf{x}(k) \notin \mathcal{X}_{j^\star(k)}$,

$$\begin{cases} \overline{\mathbf{P}}_i(k) = \mathbf{P}_i^-(k) \\ \overline{\mathbf{x}}_i(k) = \hat{\mathbf{x}}_i^-(k) \end{cases}, \quad \forall i \in \mathcal{I}, \tag{14}$$

holds. Noting that Eqs. (12)—(14) hold for all $i \in \mathcal{I}$, so that they can be made node-independent by dropping index i, that Eqs. (13)–(14) are mutually exclusive at any given time instant k, and that they can be unified thanks to the indicator function $\mathbb{1}_{\mathcal{X}_{j^\star(k)}}(\mathbf{x}(k))$, the theorem is proved. □

3. Application: Distributed Target Tracking

In this section, we apply DKNS to the *distributed tracking* of a maneuvering target performed by a network of heterogeneous range-bearing sensors with limited sensing capability. To this aim, we specialise to the case of interest some of the equations concerning the sensing process.

As it will be shown, different sensing ranges characterise the heterogeneity of the sensor network. Due to the limited sensing capability, only a subset of nodes can sense the target during a given time interval, and some uncovered areas of the space may even exist. The aim of distributed target tracking is to estimate and track the target state in the environment in discrete time, by using a distributed algorithm involving message-passing between one-hop nodes of a sensor network. Here, we illustrate the details of the distributed tracking problem formulation, providing suitable models for the environment, the target, and the sensor nodes. Then, we assess numerically the performance of the proposed approach, and we make a comparison between DKNS applied to target tracking, the Centralised Kalman Filter with multiple measurements (CKF), and two more target tracking algorithms based on distributed Kalman filtering (Cattivelli and Sayed (2010), Olfati-Saber (2007)).

3.1. *Environment, Target, and Network Model*

The network and the target lay on a two-dimensional environment $E \triangleq [-L/2, L/2] \times [-L/2, L/2] \subset \mathbb{R}^2$, $L > 0$, that is a square field with side length L. An earth-fixed Cartesian coordinate system $\{X, Y\}$, with unit vectors $\{\hat{\imath}, \hat{\jmath}\}$, respectively, is used to locate points in E. The target describes a *trajectory* $\boldsymbol{\xi}(s) \in E$ where $\boldsymbol{\xi}(s)$ are the Cartesian coordinates of the target position at the continuous time $s \in \mathbb{R}^+$. The estimates of the target state are carried out in discrete time, so that a sampling $\boldsymbol{\xi}(k) = [\xi^{[1]}(k), \xi^{[2]}(k)]^T \in E$, $k \in \mathbb{N}_0$, is performed by the network nodes. The discrete time index k is associated to a constant sampling time ε. Moreover, we denote with the vector $\boldsymbol{v}(k) = [v^{[1]}(k), v^{[2]}(k)]^T \in \mathbb{R}^2$ the target velocity at the discrete time k. Finally, the state vector to be estimated is defined as $\mathbf{x}(k) = [\xi^{[1]}(k), v^{[1]}(k), \xi^{[2]}(k), v^{[2]}(k)]^T$.

A network of n sensors is deployed in the environment E, so that each sensor is located at

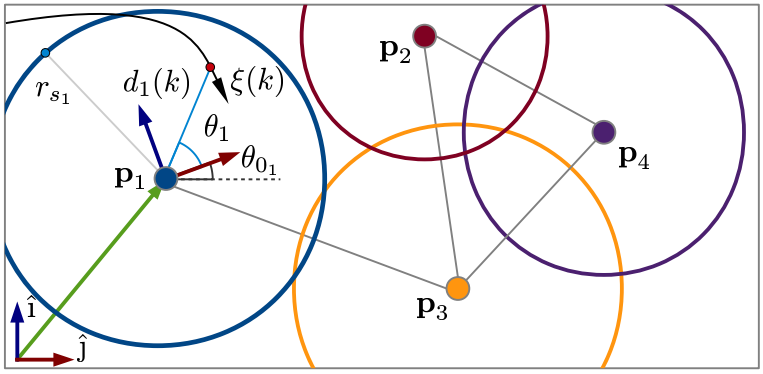


Figure 1. Abstract representation of the distributed target tracking scenario.

the position $\mathbf{p}_i = [p_i^{[X]}\, p_i^{[Y]}]^T \in E$. Sensors are connected through an undirected communication graph $\mathcal{G} = \{\mathcal{I}, \mathcal{E}\}$, with node set $\mathcal{I} = \{1, \ldots, n\}$. The edge set $\mathcal{E}$ is determined by a communication radius, r_c, identical for each node, so that $\mathcal{E} = \{(i,j) \,|\, ||\mathbf{p}_i - \mathbf{p}_j|| \leq r_c\}$. This is also known as a r_c-disk communication graph (Bullo et al. (2009)). Moreover, we assume that the graph $\mathcal{G}$ is connected and has diameter D. Finally, the limited sensing capability of each sensor is expressed via a suitable *sensing radius,* r_{s_i}. A sensor can perform a direct measurement of the target state if and only if the target is located within its sensing range, so that the *sensing set* $\mathcal{X}_i$ of the generic node i is given by

$$\mathcal{X}_i = \left\{\mathbf{x} \in \mathbb{R}^4 \mid ||\boldsymbol{\xi} - \mathbf{p}_i|| \leq r_{s_i}, \forall\, \mathbf{v}\right\}, \quad \forall i \in \mathcal{I}. \tag{15}$$

Please note that the sensing radius, and consequently its related sensing set, are considered constant in time for ease of presentation. Nevertheless, as proved in Section 2, the approach can be extended to time-varying quantities. Figure 1 shows an abstract representation of the sensor network in the case of four nodes.

3.2. *Sensor Network Node Model*

As stated above, we consider range-bearing sensors with limited sensing capability. Range-bearing sensors are able to perform direct measurements of the target distance, $d_i(k)$, and bearing, $\theta_i(k)$, with respect to a sensor-fixed reference coordinate system. We assume that each sensor node is aware of its position $\mathbf{p}_i$ and orientation θ_{0_i}, with respect to the earth-fixed coordinate frame, so that it is able to provide an estimate of the target position with respect to the latter frame through a simple coordinate transformation.

Range-bearing measurements are affected by noise, which we assume to be white Gaussian with zero mean and covariance matrix $\mathbf{B}_i(k)$. Assuming $\eta_i(k) = [d_i(k)\ \theta_i(k)]^T$, the related covariance matrix is given by (Anderson and Moore (1979)):

$$\mathbf{B}_i(k) = E[\eta_i(k)\eta_i(k)^T] = \begin{bmatrix} \sigma^2_{d_i}(k) & 0 \\ 0 & d_i^2(k)\sigma^2_{\theta_i}(k) \end{bmatrix}.$$

In range-bearing measurements, the noise grows with the distance of the target from the sensor. Therefore, to consider a realistic sensor model, a non constant variance is used. In Burguera et al. (2009), it is shown that for range-bearing sensors the variance has an almost constant trend at small distance, whereas it increases exponentially when approaching the maximum measurable value. We refer to the experimental data reported in Burguera et al. (2009), and

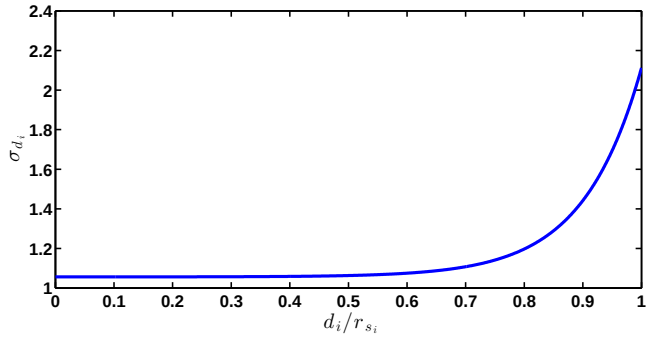


Figure 2. Trend of the standard deviation of the measurement noise σ_{d_i}, for a generic node i, versus the distance from the target, as modeled by Eq. (16). The values on the abscissa are normalised with respect to the value of the sensing range (r_{s_i}).

select the interpolation functions

$$\sigma_{d_i}(k) = k_d \left(1 + e^{k_r\left[(d_i(k) - r_{s_i})/r_{s_i}\right]}\right), \quad d_i(k) \leq r_{s_i}, \tag{16}$$

and

$$\sigma_{\theta_i}(k) = k_\theta \frac{d_i(k)}{r_{s_i}}, \qquad d_i(k) \leq r_{s_i}, \tag{17}$$

whose trends are suitable to fit the provided experimental data. It is clear that the values of $\sigma_{d_i}(k)$ and $\sigma_{\theta_i}(k)$ are not defined for $d_i(k) > r_{s_i}$. Figure 2 shows the trend of $\sigma_{d_i}(k)$, as expressed in Eq. (16).

According to the range-bearing model (Anderson and Moore (1979)), we assume that a sensing node (*i.e.*, $i \in \mathcal{S}(k)$) is able to compute the position of the target with respect to the earth-fixed Cartesian coordinate frame, as

$$\mathbf{z}_i(k) = [p_i^{[X]} + d_i(k)\cos(\theta_{0_i} + \theta_i(k)),\ p_i^{[Y]} + d_i(k)\sin(\theta_{0_i} + \theta_i(k))]^T. \tag{18}$$

Keeping in mind that each node runs a local Kalman filter, and assuming that sensing nodes cannot measure directly the target velocity, we formalise the observation equation (18) as

$$\mathbf{z}_i(k) = \mathbf{H}\mathbf{x}(k) + \boldsymbol{\nu}_i(k), \tag{19}$$

where $\mathbf{H}$ is the measurement output matrix, defined as:

$$\mathbf{H} = \begin{bmatrix} 1\,0\,0\,0 \\ 0\,0\,1\,0 \end{bmatrix},$$

and $\boldsymbol{\nu}_i(k) \in \mathbb{R}^2$ is the measurement noise, for which each component is assumed to be white Gaussian with zero mean and covariance matrix $\mathbf{R}_i(k)$, which is given by (Anderson and Moore (1979))

$$\mathbf{R}_i(k) = \mathbf{T}_i(k)\mathbf{B}_i(k)\mathbf{T}_i^T(k),$$

where $\mathbf{T}_i(k)$ is the rotation matrix

$$\mathbf{T}_i(k) = \begin{bmatrix} \cos(\theta_i(k)) & -\sin(\theta_i(k)) \\ \sin(\theta_i(k)) & \cos(\theta_i(k)) \end{bmatrix}.$$

The dynamical model of the target motion, embedded in each node's Kalman filter, is defined as:

$$\mathbf{x}(k) = \mathbf{A}\mathbf{x}(k-1) + \boldsymbol{\omega}(k), \tag{20}$$

where

$$\mathbf{A} = \mathbf{I}_2 \otimes \begin{bmatrix} 1 & \varepsilon \\ 0 & 1 \end{bmatrix},$$

ε is the time step, $\otimes$ is the Kronecker product of matrices, and $\boldsymbol{\omega}(k) \in \mathbb{R}^4$,

$$\boldsymbol{\omega}(k) = \begin{bmatrix} \omega^{[1]}(k) & \omega^{[2]}(k) & \omega^{[3]}(k) & \omega^{[4]}(k) \end{bmatrix}^T,$$

is a noise term that is white Gaussian with zero mean and covariance matrix $\mathbf{Q}$, *i.e.,* $\boldsymbol{\omega}(k) \sim N(0, \mathbf{Q})$. Here, we assume

$$\omega^{[1]}(k) = \tfrac{\varepsilon}{2}\omega^{[2]}(k), \quad \omega^{[3]}(k) = \tfrac{\varepsilon}{2}\omega^{[4]}(k), \tag{21}$$

to take into account the influence of the maneuvering target on the positional coordinates (Anderson and Moore (1979)). Thus, matrix $\mathbf{Q}$, is given by

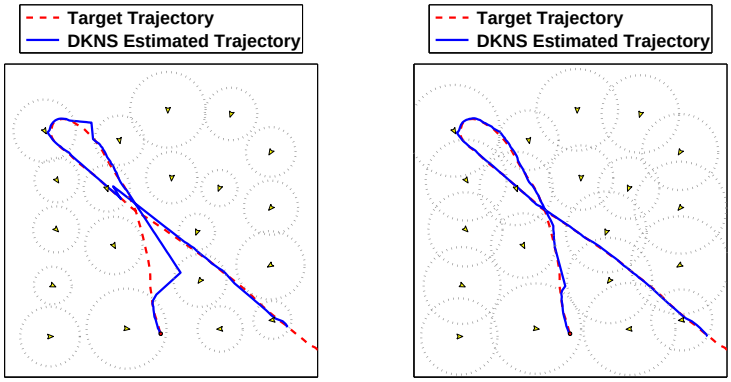
$$\mathbf{Q} = \begin{bmatrix} \frac{\varepsilon^2}{4}\sigma^2_{\omega^{[2]}} & \frac{\varepsilon}{2}\sigma^2_{\omega^{[2]}} & 0 & 0 \\ \frac{\varepsilon}{2}\sigma^2_{\omega^{[2]}} & \sigma^2_{\omega^{[2]}} & 0 & 0 \\ 0 & 0 & \frac{\varepsilon^2}{4}\sigma^2_{\omega^{[4]}} & \frac{\varepsilon}{2}\sigma^2_{\omega^{[4]}} \\ 0 & 0 & \frac{\varepsilon}{2}\sigma^2_{\omega^{[4]}} & \sigma^2_{\omega^{[4]}} \end{bmatrix}, \tag{22}$$

where $\sigma^2_{\omega^{[2]}}$ and $\sigma^2_{\omega^{[4]}}$ are the variances of the stochastic processes $\omega^{[2]}(k)$ and $\omega^{[4]}(k)$, respectively.

3.3. *Tracking Setup*

The performance of the DKNS algorithm is analysed by running a campaign of Monte Carlo simulations. Each set of experiments consists of the tracking of 100 repeated random target trajectories. With the aim of comparing our approach with KCF and DKF (Cattivelli and Sayed (2010), Olfati-Saber (2007)), we simulate a sensor network tracking the position of a maneuvering target moving inside a square field E with side length $L = 90$ and with a communication radius set as $r_c = 3\frac{L}{\lceil\sqrt{n}\rceil+1} + 2$. The nodes are placed at random positions in the environment, yet preserving connectedness of the network. In order to evaluate more in depth the performance of the DKNS, we define the percentage sensing coverage ratio ρ, that is the ratio between the sensing area covered by all nodes, and the total area of the field E,

$$\rho = \frac{\mathcal{A}(\bigcup_{i=1}^{n} \mathcal{X}_i)}{\mathcal{A}(E)} \cdot 100\,, \tag{23}$$



(a) (b)

Figure 3. DKNS simulations of a random generated target trajectory for a network of $n = 20$ heterogeneous nodes. The target comes from the bottom right corner of the environment, and the filled circle indicates its last position. (a) A simulation of the target tracking with $\rho = 45\%$. (b) The tracking of the same target trajectory with $\rho = 78\%$. In the latter case, due to the increasing coverage ratio ρ, the tracking accuracy α is higher than in the former.

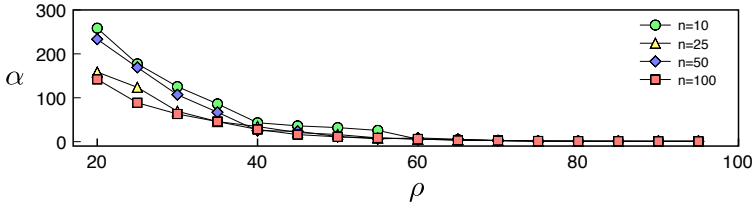
where $\mathcal{A}(\cup_{i=1}^{n}\mathcal{X}_i)$ is the area of the union of all $\mathcal{X}_i$ sensing sets, and $\mathcal{A}(E) = L^2$, is the area of the environment. Given the random distribution of the positions of the sensor nodes in the environment, we achieve a desired coverage ratio ρ by setting at first the sensing radius r_{s_i}, for each node $i \in \mathcal{I}$, to a suitable value r_ρ, computed to guarantee the coverage ratio ρ. A suitable algorithm to find r_ρ is adopted, which takes into account the existence of overlapping sensing areas. Then, heterogeneity in the sensor network is achieved by adding to the sensing radii r_{s_i}, for each node $i \in \mathcal{I}$, a Gaussian noise with zero mean and standard deviation $\sigma_\rho = 0.03 r_\rho$. The initial guess of the estimated target state is $\overline{\mathbf{x}}_i(0) = \mathbf{0}$, $\forall i \in \mathcal{I}$. Moreover, the variances $\sigma_{\omega^{[2]}}$ and $\sigma_{\omega^{[4]}}$ of the process noise covariance matrix of the Kalman filter in Eq. (22), are chosen as $\sigma_{\omega^{[2]}} = \sigma_{\omega^{[4]}} = 3$, $\forall i \in \mathcal{I}$. The variances related to the range-bearing sensing process, $\sigma_{d_i}(k)$ and $\sigma_{\theta_i}(k)$, are modeled by Eqs. (16-17), setting $k_d = 1.056$, $k_r = 10.07$ and $k_\theta = 0.1$, in order to fit the experimental dataset in Burguera et al. (2009). Finally, again for comparison purposes we set $\mathbf{P}(0) = 10\sigma_0^2\mathbf{I}_4$, where $\sigma_0 = 5$.

Among the several aspects regarding performance assessment in target tracking applications (Zhao and Guibas (2004)), in this work we will focus on *tracking accuracy,* by evaluating the mean square error between estimated and actual target trajectory. As a metric for target tracking accuracy, the following mean square error (in norm) is computed:

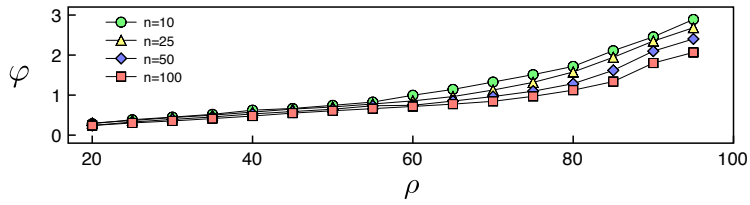
$$\alpha = \frac{1}{k_f}\sum_{k=1}^{k_f} \|\mathbf{H}\,\overline{\mathbf{x}}_i(k) - \boldsymbol{\xi}(k)\|^2 \tag{24}$$

where k_f is the duration (in time samples) of the target trajectory, $\boldsymbol{\xi}(k)$ is the actual target position at discrete time k, $\mathbf{H}\,\overline{\mathbf{x}}_i(k)$ is any of the global target position estimates $\overline{\mathbf{x}}_i(k)$ at time k. We remind that, as proved in Theorem 2.2, the global target position estimates are identical for each network node. Obviously, the ideal condition of perfect tracking is $\alpha = 0$, and the tracking accuracy is better as long as α is smaller.

Furthermore, we define a third index, φ, which is the average percentage of sensing nodes



(a)



(b)

Figure 4. Performance evaluation of DKNS algorithm. Values of α (a) and φ (b) as function of the coverage ratio ρ for networks with $n = 10$, $n = 25$, $n = 50$, and $n = 100$ nodes. Each point is computed by averaging 100 random trajectories.

during a single run. It is defined as

$$\varphi = \frac{1}{k_f} \sum_{k=1}^{k_f} \frac{|\mathcal{S}(k)|}{n} \cdot 100\,, \tag{25}$$

where $|\cdot|$ is the set cardinality operator.

3.4. *DKNS Performance Evaluation*

Two single experiments of the aforementioned simulation campaign are illustrated in Figure 3. In particular, the same trajectory is estimated via the DKNS algorithm by a network of $n = 20$ nodes, under two different coverage ratio conditions, and the estimated trajectory performed by the sensor network is plotted against the one of the target. It is important to note that, keeping constant the number of sensor nodes n, the performance improves as long as the coverage ratio ρ increases. As an example, in Figure 3(a), a tracking accuracy $\alpha = 42.81$ is obtained with a coverage ratio $\rho = 45\%$, while in Figure 3(b) a tracking accuracy $\alpha = 2.47$ is obtained with a coverage percentage of $\rho = 78\%$.

It can be noted from Figure 3, that when the target is not sensed by any node, the estimate is performed only through the linear model defined in Eq. (8). Then, when the target comes back into the sensing set of any sensor, the estimate is performed by exploiting the whole Kalman filter, thus improving the corresponding estimate.

An extensive evaluation of the performance of DKNS, made on the entire simulation campaign, is illustrated in Figure 4. In Figure 4(a), the tracking accuracy α is plotted versus the coverage ratio ρ, for different values of the number of nodes n. As can be noted, the tracking accuracy improves as long as the coverage ratio ρ increases. Furthermore, it can be observed that the performance tends to be independent from the number of nodes when the coverage ratio ρ grows (in Figure 4(a), the performance is practically independent from the number of sensors for $\rho > 60\%$). It can also be noted that the same tracking accuracy α can be obtained for different pairs of the number of nodes n and of the coverage ratio ρ. For example, the performance index

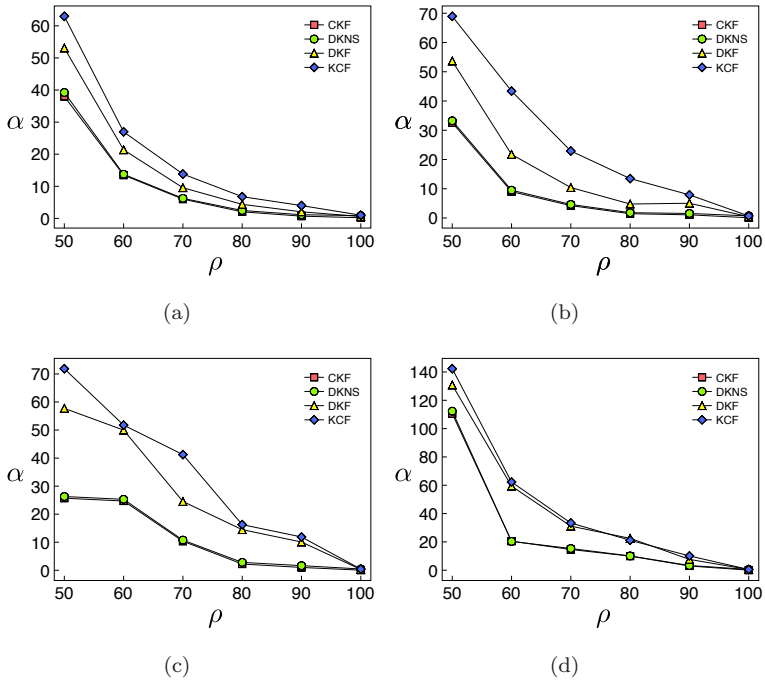


Figure 5. Comparison between DKNS and three other tracking algorithms: CKF, KCF, and DKF. Values of α as function of the coverage ratio ρ for networks with $n = 10$ (a), $n = 15$ (b), $n = 25$ (c), and $n = 50$ nodes (d). Each point is computed by averaging 100 random trajectories.

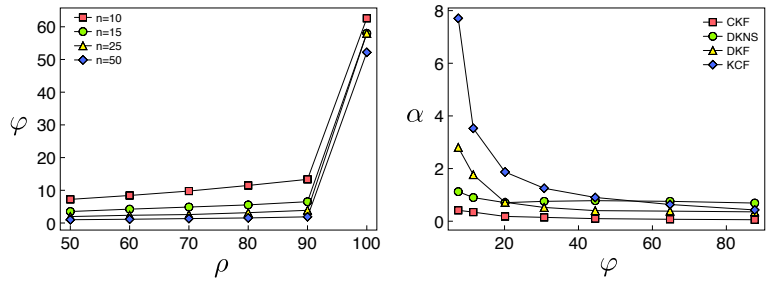
$\alpha = 100$ is achieved with $(n = 10, \rho = 33.82\%)$, $(n = 25, \rho = 27.52\%)$, $(n = 50, \rho = 30.73\%)$, $(n = 100, \rho = 24.12\%)$. From the simulation results, it is evident that to guarantee a desired accuracy, the number of network nodes and the coverage ratio are of fundamental importance. Nevertheless, one can make up for a small number of sensor nodes via the setting of a suitable coverage ratio, obtained both an effective spatial deployment, and by setting large enough sensing radii. The last consideration can help in overcoming, in real-time applications, the growth of the convergence time of the max-consensus algorithm, which increases with the diameter D of the network, that can be influenced by the number of nodes n (See Section 2.3).

Figure 4(b) illustrates the average number (in percentage) of sensing nodes φ during a run of the algorithm, versus the coverage ratio ρ, for different values of the number of nodes n.

3.5. *Comparison with other Target Tracking Algorithms based on Distributed Kalman Filtering*

In this section, we compare the performance of the DKNS algorithm with a centralised and two distributed target tracking approaches based on Kalman filtering. The first algorithm is the Centralised Kalman filter with multiple measurements (CKF) (Mitchell (2007)), which is known to be optimal in the sense of the minimization of the error variance, thus exhibiting the best performance. The second algorithm is the Kalman Consensus Filter (KCF) with message passing (Algorithm 3 in Olfati-Saber (2007)), and the third one is the Diffusion Kalman Filter (DKF), presented by Cattivelli and Sayed (2010).

The comparison is performed through 100 repeated random trajectories for different values of the coverage ratio ρ, and of the number of sensor nodes, n. More specifically, the average value of α over the 100 test trajectories has been computed for each value of ρ, and compared with the corresponding performance parameter obtained through the CKF, KCF, and DKF algorithms, run on the same test trajectories. Figure 5 shows that the DKNS algorithm generally



(a) (b)

Figure 6. (a) Values of φ as function of the coverage ratio ρ for networks with $n = 10, 15, 25, 50$ nodes. (b) Values of α as function of the index φ for a network of $n = 15$ nodes.

outperforms KCF and DKF and, for a given value of the coverage ratio ρ, the gap in performance generally increases as long as the number of network nodes increases. Nevertheless, the gap in performance decreases as long as the coverage ratio ρ increases. It is also important to note that DKNS closely approaches the optimal performance of the CKF, as long as the number of network nodes increases. The explanation of this behaviour resides in the trend of the average percentage of sensing nodes φ, illustrated in Figure 6(a). In fact, keeping constant the coverage ratio ρ, and increasing the number of network nodes, yields a decrease of the average number of sensing nodes φ. Hence, it can be noted that the lower is φ, the better is the approximation of the centralised tracking exhibited by DKNS. This is well explained by considering that the one-term approximation of Eq. (6), expressed in Eq. (10), is better as long as the number of sensing nodes is smaller. This statement is supported by the numerical results illustrated in Figure 6(b), where it can be observed that the performance of DKNS is quite independent from the value of φ, which is not the case when KCF and DKF are considered. Moreover, it is evident that the application of DKNS is not the best option in any case: in fact, when the average number of sensing nodes φ grows, KCF and DKF tend to perform better than DKNS. In our simulations, this happens for $\varphi > 20\%$ for DKF, and for $\varphi > 57\%$ for KCF.

4. Conclusions

In this paper, we have addressed the problem of distributed Kalman filtering over heterogeneous sensor networks, by introducing a novel approach, called Distributed Kalman filtering with Node Selection (DKNS). This is proved to be equivalent to a centralised Kalman filter with multiple measurements, which evolves according to a state-dependent switching dynamics, able to select and propagate the best estimate of the process state through the sensor network in finite time. The optimality of the estimate accuracy is assessed through the definition of a metric, called perception confidence value, which is strictly related to the Fisher information.

DKNS is particularly suitable in case of sensor networks with limited sensing capability, that is, in those cases in which only a few sensors in the network can actually perform a direct measurement of the process state. In these cases, in fact, fusing information may often lead to poor results, whereas node selection may constitute a better option. Conversely, when many reliable measurements are available, information fusion may result in better performance than node selection.

We have applied the algorithm to the discrete-time tracking of a maneuvering target, performed by a network of heterogeneous range-bearing sensors with limited sensing capability, achieving very satisfactory results. The performance comparison with existing algorithms based on sensor fusion reflects what mentioned above in general terms, that is, DKNS is a viable and more

effective option as long as limitations on the sensing capability are present.

Further work will cope with the possible presence of malicious or misbehaving nodes. In fact, if a generic node of the network, either deliberately or due to a fault, provides an inaccurate prediction, while exhibiting a high perception confidence value, then the tracking algorithm will converge to an estimate which may not be optimal. Moreover, future research directions in the study of DKNS will deal with the removal of the hypothesis concerning the synchronicity of operations. Interesting research paths are also envisioned when the selection of more than one node, or the execution of the agreement phase for a number of iterations less than that needed for the convergence of the max-consensus protocol, are considered. Finally, on the application side, our plan is to extend the distributed target tracking algorithm based on DKNS through the inclusion of sensor nodes carried by mobile agents, in order to improve the estimation performance (consequently, a time-varying topology of the network will be considered).

Acknowledgements

We would like to thank Paolo Da Pelo for inspiring discussions and Grazia Cicirelli for contributions to the early idea of this approach.

References

Anderson, B.D.O., and Moore, J.B., *Optimal Filtering*, Thomas Kailath Editor (1979).

Bondy, A., and Murty, U., *Graph Theory*, Graduate Texts in Mathematics, Springer (2008).

Bullo, F., Cortés, J., and Martínez, S., *Distributed Control of Robotic Networks*, Electronically available at http://coordinationbook.info, Applied Mathematics Series, Princeton University Press (2009).

Burguera, A., González, Y., and Oliver, G. (2009), "Sonar Sensor Models and Their Application to Mobile Robot Localization," *Sensors*, 12(9), 10217–10243.

Buscarino, A., Fortuna, L., Frasca, M., and Rizzo, A. (2006), "Dynamical Network Interactions in Distributed Control of Robots," *Chaos*, 16(1), 015116.

Buscarino, A., Fortuna, L., Frasca, M., and Rizzo, A. (2007), "Effects of Long-Range Connections in Distributed Control of Collective Motion," *International Journal of Bifurcation and Chaos*, 17(7), 2411–2417.

Cao, Y., and Ren, W. (2010), "Multi-Agent Consensus Using Both Current and Outdated States with Fixed and Undirected Interaction," *Journal of Intelligent and Robotic Systems*, 58, 95–106.

Cardoso, J.C.S., Baquero, C., and Almeida, P.S. (2009), "Probabilistic Estimation of Network Size and Diameter Dependable Computing," in *4th Latin-American Symposium on Dependable Computing (LADC)*, Seville, Spain, pp. 33–40.

Cattivelli, F., and Sayed, A. (2010), "Diffusion Strategies for Distributed Kalman Filtering and Smoothing," *IEEE Transactions on Automatic Control*, 55(9), 2069–2084.

Dargie, W., and Poellabauer, C., *Fundamentals of Wireless Sensor Networks: Theory and Practice*, Wiley Series on Wireless Communication and Mobile Computing, Wiley (2010).

Di Paola, D., Naso, D., and Turchiano, B. (2011), "Consensus-based Robust Decentralized Task Assignment for Heterogeneous Robotic Networks," in *American Control Conference (ACC)*, June, pp. 4711–4716.

Fax, A., and Murray, R.M. (2004), "Information flow and cooperative control of vehicle formations," *IEEE Transactions on Automatic Control*, 49(9), 1465–1476.

Ghaffarkhah, A., and Mostofi, Y. (2009), "Communication-Aware Target Tracking using Navi-

gation Functions - Centralized Case," in *Robot Communication and Coordination Conference, 2009 International Conference on*, April, Odense, Denmark, pp. 1–8.

Giannini, S., Petitti, A., Di Paola, D., and Rizzo, R. (2013), "Consensus-Based Distributed Target Tracking with Asynchronous Measurements," in *IEEE International Conference on Decision and Control (CDC)*, Dec., Florence, Italy.

Grocholsky, B., Makarenko, A., and Durrant-Whyte, H. (2003), "Information-Theoretic Coordinated Control of Multiple Sensor Platforms," in *IEEE International Conference on Robotics and Automation (ICRA)*, Vol. 1, pp. 1521–1526.

Kalman, R.E. (1960), "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D), 35–45.

La, H.M., and Sheng, W. (2012), "Dynamic Target Tracking and Observing in a Mobile Sensor Network," *Robotics and Autonomous Systems*, 60(7), 996 – 1009.

Mahmoud, M., and Khalid, H. (2013), "Distributed Kalman filtering: a Bibliographic Review," *Control Theory Applications, IET*, 7(4), 483–501.

Martinez, S., Cortes, J., and Bullo, F. (2005), "On Robust Rendezvous for Mobile Autonomous Agents," in *Proceedings of the 16th IFAC World Congress*, July, Prague, Czech Republic.

Martinez, S., and Bullo, F. (2006), "Optimal Sensor Placement and Motion Coordination for Target Tracking," *Automatica*, 42(4), 661–668.

Millán, P., Orihuela, L., Jurado, I., Vivas, C., and Rubio, F.R. (2013), "Distributed Estimation in Networked Systems Under Periodic and Event-based Communication Policies," *International Journal of Systems Science*, In Press.

Mitchell, H., *Multi-sensor Data Fusion: An Introduction*, 1st ed., Springer Publishing Company, Incorporated (2007).

Nejad, B.M., Attia, S.A., and 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 (ICAT)*, October, Bosnia, pp. 1–7.

Olfati-Saber, R. (2006), "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Transactions on Automatic Control*, 51(3), 401–420.

Olfati-Saber, R. (2007), "Distributed Kalman Filtering for Sensor Networks," in *IEEE International Conference on Decision and Control Conference (CDC)*, Dec., New Orleans, LA, USA, pp. 5492–5498.

Olfati-Saber, R., Fax, A., and Murray, R.M. (2007), "Consensus and Cooperation in Networked Multi-Agent Systems," *Proceedings of the IEEE*, 95(1), 215–233.

Oruc, S., Sijs, J., and van den Bosch, P. (2009), "Optimal Decentralized Kalman Filter," in *Mediterranean Conference on Control and Automation Conference (MED)*, June, Thessaloniki, Greece, pp. 803–808.

Petitti, A., Di Paola, D., Rizzo, A., and Cicirelli, G. (2011), "Consensus-based Distributed Estimation for Target Tracking in Heterogeneous Sensor Networks," in *IEEE International Conference on Decision and Control and European Control Conference (CDC-ECC)*, Dec., pp. 6648 –6653.

Rao, B.S.Y., Durrant-Whyte, H.F., and Sheen, J.A. (1993), "A Fully Decentralized Multi-Sensor System for Tracking and Surveillance," *International Journal of Robotics Research*, 12(1), 20–44.

Ren, W., Beard, R.W., and Atkins, E.M. (2005), "A Survey of Consensus Problems in Multi-agent Coordination," in *American Control Conference (ACC)*, June, Portland, OR, USA, pp. 1859–1864.

Ren, W., Beard, R.W., and Atkins, E.M. (2007), "Information Consensus in Multivehicle Cooperative Control," *IEEE Control System Magazine*, 27(2), 71–82.

Ren, W., and Beard, R.W., *Distributed Consensus in Multi-vehicle Cooperative Control: Theory and Applications (Communications and Control Engineering)*, 1 ed., Springer (2007).

Siouris, G.M., *An Engineering Approach to Optimal Control and Estimation Theory*, Wiley-

Interscience Publication (1979).

Taj, M., and Cavallaro, A. (2009), "Multi-Camera Track-Before-Detect," in *International Conference on Distributed Smart Cameras (ICDSC)*, August, Como, IT.

Wang, C.L., and Wu, D.S. (2008), "Decentralized Target Tracking Based on a Weighted Extended Kalman Filter for Wireless Sensor Networks," in *IEEE International Conference on Global Telecommunications (GLOBECOM)*, Dec., New Orleans, LO, pp. 1–5.

Yang, W., and Shi, H. (2012), "Sensor Selection Schemes for Consensus Based Distributed Estimation Over Energy Constrained Wireless Sensor Networks," *Neurocomputing*, 87, 132–137.

Yang, W., Wang, X., and Shi, H. (2011), "Optimal Consensus-based Distributed Estimation with Intermittent Communication," *International Journal of Systems Science*, 42(9), 1521–1529.

Yu, H., Zhuang, Y., and Wang, W. (2011), "Distributed H-∞ Filtering with Consensus in Sensor Networks: a Two-dimensional System-based Approach," *International Journal of Systems Science*, 42(9), 1543–1557.

Zhao, F., and Guibas, L., *Wireless Sensor Networks: An Information Processing Approach*, The Morgan Kaufmann Series in Networking, Morgan Kaufmann (2004).