VERSITA

Int. J. Appl. Math. Comput. Sci., 2013, Vol. 23, No. 2, 327–339
DOI: 10.2478/amcs-2013-0025

amcs

ASYNCHRONOUS DISTRIBUTED STATE ESTIMATION FOR CONTINUOUS–TIME STOCHASTIC PROCESSES

ZDZISŁAW KOWALCZUK, MARIUSZ DOMŻALSKI

Department of Decision Systems (ETI)
Gdańsk University of Technology, Narutowicza 11/12, 80-233 Gdańsk, Poland
e-mail: {kova,mardo}@eti.pg.gda.pl

The problem of state estimation of a continuous-time stochastic process using an Asynchronous Distributed multi-sensor Estimation (ADE) system is considered. The state of a process of interest is estimated by a group of local estimators constituting the proposed ADE system. Each estimator is based, e.g., on a Kalman filter and performs single sensor filtration and fusion of its local results with the results from other/remote processors to compute possibly the best state estimates. In performing data fusion, however, two important issues need to be addressed namely, the problem of asynchronism of local processors and the issue of unknown correlation between asynchronous data in local processors. Both the problems, along with their solutions, are investigated in this paper. Possible applications and effectiveness of the proposed ADE approach are illustrated by simulated experiments, including a non-complete connection graph of such a distributed estimation system.

Keywords: continuous-time stochastic processes, distributed systems, state estimation, Kalman filtering.

1. Introduction

The main objective of estimation is to infer on the state of an observed process of interest. Classical estimation is based on Kalman filtering (Kalman, 1960; Bar-Shalom and Li, 1993; Blackman and Popoli, 1999), usually processing data from a single data source at predefined time moments. Precision and robustness of the classical state estimation based on one sensor can, in certain circumstances, appear unacceptable. Multiple-data sources can be considered in such cases.

Distributed (decentralized) estimation and identification (Hall and Llinas, 1997; Liggins *et al.*, 1997; Ribeiro *et al.*, 2006; Uciński, 2012; Patan, 2012) is an important and challenging issue in the field of control and estimation. A typical multi-sensor distributed estimation system is composed of a set of local data processors, also referred to as nodes, accordingly connected. Each of those local processors, computing state estimates of the observed process, works on data obtained from two groups of sources. The first group of sources is composed of sensors integrated with a given local processor. The second group is comprised of the remaining (local otherwise) processors working within the same system. This means that each local processor combines data supplied by its own sensors and other processing nodes. Such a process is called data fusion.

Two sample configurations of a distributed system, composed of four processing nodes (working as local data processors), are presented in Fig. 1. In these configurations each node processes data from one sensor and data gained from the other three nodes (the dashed line represents a border between the two groups for node 1). However, in the configuration presented in Fig. 1(a) each node is connected with all other nodes (i.e., the system has a complete connection graph), whereas in the configuration of Fig. 1(b) the nodes are interconnected in a chain form (the system has thus an incomplete connection graph).

The main advantages of such a completely distributed processing system lie in its robustness and flexibility. Any failure in any local processor does not stop the other processing nodes from functioning. Moreover, models of the process used in each local processor can be completely different. The models can, for example, be chosen specifically (optimally) to cope with data taken from the process observed by a given local sensor. What is more, as the data passed between the local systems are of the same type (state estimates), the existing infrastructure of the distributed estimation system can be easily extended to compose a new system structure or to include new processors in an existing

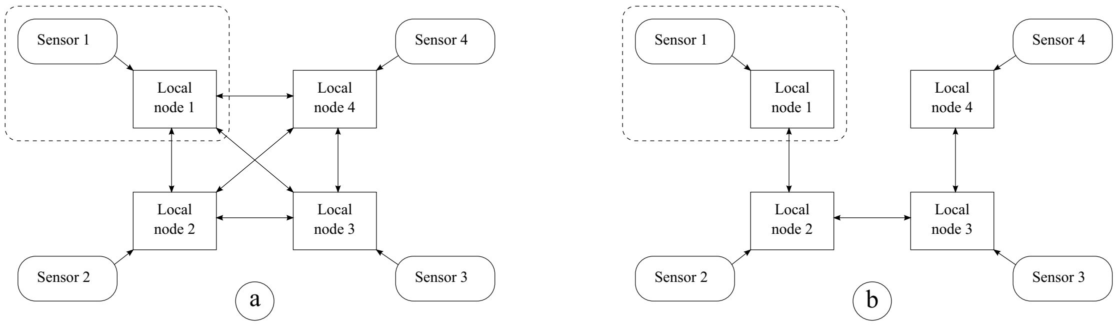

Fig. 1. Distributed estimation system: complete connection graph (a), incomplete connection graph (b).

system. Thus decentralization can effectively provide a new degree of scalability in estimation systems. Note that typical applications of such distributed estimation systems can be found both in air traffic control and in mobile robotics.

In the recent years two groups of distributed estimation algorithms for dynamic systems have emerged. The first group can be associated with average consensus algorithms. Solutions for distributed estimation of scalar random processes based on average consensus over a network of filters is presented by Olfati-Saber and Shamma (2005), Olfati-Saber (2007), Xiao *et al.* (2005) and Speranzon *et al.* (2006), and is refined by Del Favero and Zampieri (2009) for the case of communication in a sensor network according to randomized strategies (in each cycle a pair of nodes is allotted for mutual data exchange). Distributed estimation of states of linear dynamic systems is discussed by Carli *et al.* (2008) as well as Cattivelli and Sayed (2010). In distributed systems based on the average consensus approach, at each time step, each node takes a weighted average of differences between the values received from the other nodes and the value of its own estimate. If this average is positive, the estimate for this particular node is increased proportionally to the average, and if this average is negative, the estimate is decreased. Both the amount of information exchanged between the nodes and the weights used in the average consensus algorithms, which can be chosen according to many different strategies, affect the rate of convergence toward the asymptotic agreement between the nodes.

The distributed method presented in this paper belongs to the second group of estimation algorithms that are based on a set of state estimators exchanging full state information between themselves (Rao *et al.*, 1993; Bar-Shalom and Li, 1995; Hall and Llinas, 2001). This information is usually comprised of current state estimates and their corresponding covariance matrices. In each of the computing nodes, weights used for data fusion of local and foreign/received estimates are computed based on these covariance matrices and are therefore proportional to the quality of particular estimates.

There is an essential assumption in the consensus filter that all the nodes use the same process model. Whereas in our system comprised of the local state estimators each node can utilize a different model for the observed process (assuming that the set of state variables is the same for each model, or the state variables can be transformed from one set to another; otherwise the fusion, or rather an aggregation, has to be performed in a completely different way).

Unfortunately, distributed estimation brings about certain difficulties. One problem is unknown cross-correlation between data in local processors. Even with the assumption that measurements from sensors are uncorrelated, data in the processing nodes are certain to be correlated. In particular, data received by a local processor from other nodes contain information spread by this processor over the network in previous cycles. Such information (used more than once) corrupts the results of state estimation. It is clear that cross-correlation cannot be ignored and has to be taken into account in a data fusion process in a way that the resulting state estimates stay consistent. A solution to the problem of unknown cross-correlation is incorporated in a method based on a set of estimators (Julier and Uhlmann, 1997). This problem is rarely addressed in the context of consensus filters.

Another important issue is that processing nodes can work asynchronously, as they can perform measurements and compute state estimates at different time moments. However, both the average consensus methods and those based on local estimators usually assume that sensors take measurements and communicate with each other synchronously or that the sensors exchange data multiple times at a regular pace during each of measurement cycles.

In the case of truly Asynchronous Distributed Estimation (ADE), data fusion is of a special meaning, since each node of the system has to perform synchronization between its own data and the data obtained from other nodes, before fusing them.

A straightforward solution to the problem of asynchronous estimation consists in using Kalman filters and discrete time models expressed in terms of the corresponding system matrices dependent on the current sampling time (possibly different for each cycle). Such a method is presented by Bilenne (2004), for example.

Another solution to the problem of estimation based on data gained at different sampling intervals is multi-rate Kalman filtering (Chen and Chen, 1995; Kuchler and Therrien, 2003). A multi-rate Kalman filter performs state estimation based on data from multiple sources, each working at a different sampling rate. Each of these sources is described by an appropriate discrete-time model.

The main drawback of the above methods is the need for evaluation of multiple discrete-time models, where each model corresponds to a different sampling time. This is not an issue when using the methods based on continuous-time (c-t) models.

An optimal estimator for continuous-time linear stochastic systems applicable to arbitrary combination of continuous and discrete measurements, which can be sampled at unknown, variant, and possibly random rates and delays, is presented by Zhang *et al*. (2007). Such an optimal estimator is not, however, designed for decentralized implementation (the necessary transfer of c-t process trajectories along with their corresponding c-t covariance matrices between the distributed nodes of the system can be a complex task). Therefore, we propose an ADE method that is, due to its distributed nature, more robust in the face of communication and processing failures (though it yet ignores communication delays). In order to accomplish this effect, we utilize a hybrid approach in which the c-t stochastic process model is used as a basis for analyzing and solving the synchronization problem of discrete-time data transferred between the system nodes.

Nonlinear problems (interesting from several practical points of view) are not discussed in this paper. This is because the use of nonlinear models and appropriate estimation algorithms causes additional problems with respect to the sheer multi-sensor estimation problem. Thus, at this stage, it would be difficult to draw general conclusions about the discussed asynchronous distributed estimation system independent of the other issues.

The paper is organized as follows. In Section 2 a continuous-time stochastic process model and its discrete-time version are presented. Next, the problem of data synchronization is described in Section 3. A complete ADE fusion algorithm is derived in Section 4. Simulation examples and applications are presented in Section 5. Section 6 contains conclusions and plans for future research.

2. Process model

As has been mentioned above, to deal with asynchronous data a suitable continuous-time (c-t) process model (Karatzas and Shreve, 1991; Oksendal, 2003) is to be considered, whose stochastic nature allows us to aptly describe data uncertainty. A corresponding sampled-data model will be obtained afterwards.

2.1. c-t Gauss–Markov model. It is assumed that the dynamics of the process of interest can be described by the following n-dimensional linear stochastic differential equation:

$$\begin{aligned} \mathrm{d}X(t) &= [AX(t)+b]\,\mathrm{d}t + \sigma\,\mathrm{d}w(t), \quad 0 \le t < \infty, \\ X(0) &= X_0, \end{aligned} \tag{1}$$

where w is the r-dimensional Brownian motion independent of the initial vector X_0, which has a given n-variate normal distribution. The $(n \times n)$, $(n \times 1)$ and $(n \times r)$ matrices A, b, and σ, respectively, are nonrandom and bounded.

The solution of (1) has the following representation (Karatzas and Shreve, 1991; Rogers and Williams, 2000):

$$\begin{aligned} X(t) \triangleq \Phi(t)\Bigg[X(0) &+ \int_0^t \Phi^{-1}(\eta) b\,\mathrm{d}\eta \\ &+ \int_0^t \Phi^{-1}(\eta)\sigma\,\mathrm{d}w(\eta)\Bigg], \quad 0 \le t < \infty, \end{aligned} \tag{2}$$

where $\Phi(t)$ is a non singular matrix called the fundamental solution of the following homogeneous ordinary differential equation:

$$\dot{\zeta}(t) = A\zeta(t). \tag{3}$$

The fundamental solution to (3) can be easily expressed as

$$\Phi(t) = e^{tA} \triangleq \sum_{i=0}^{\infty} \frac{t^i}{i!} A^i. \tag{4}$$

It can be proved that X in (2) is a Gaussian process with the strong Markov property. Thus the finite-dimensional distributions of the process X are completely determined by the mean and the covariance functions (Karatzas and Shreve, 1991; Rogers and Williams, 2000).

2.2. Sampled-data model. As the data fusion is performed at selected discrete-time moments, a sampled-data process model is necessary. Such a model can be obtained based on the above c-t model.

For any two particular time moments τ and t, $\tau \le t$, the following transitional (reference) equation results from the c-t model described above by (1)–(4):

$$x(t) = F(t,\tau)x(\tau) + u(t,\tau) + w(t,\tau) \tag{5}$$

with

$$F(t,\tau) = \Phi(t)\Phi^{-1}(\tau), \tag{6}$$

$$u(t,\tau) = \Phi(t)\int_{\tau}^{t} \Phi^{-1}(\eta) b \,\mathrm{d}\eta, \tag{7}$$

$$w(t,\tau) = \Phi(t)\int_{\tau}^{t} \Phi^{-1}(\eta)\sigma \,\mathrm{d}w(\eta), \tag{8}$$

where (8) represents the Itô stochastic integral (Karatzas and Shreve, 1991; Rogers and Williams, 2000; Oksendal, 2003), which can be calculated by parts. The mean of (8) is $E\{w(t,\tau)\} = 0, \forall t,\tau$, and the covariance matrix of (8) is

$$\begin{aligned} Q(t,\tau) &\triangleq E\left\{w(t,\tau)\left[w(t,\tau)\right]^{\top}\right\} \\ &= \Phi(t)\left[\int_{\tau}^{t} \Phi^{-1}(\eta)\sigma\left[\Phi^{-1}(\eta)\sigma\right]^{\top}\,\mathrm{d}\eta\right]\Phi^{\top}(t). \end{aligned} \tag{9}$$

A useful optimal solution for asynchronous sampling of two dimensional Gauss–Markov stochastic processes, based on explicit forms of the matrix exponential, is presented by Kowalczuk and Domżalski (2012a).

A necessary discrete-time signal ($z_i \in \mathbb{R}^p$) of observations of the process $x(t)$, performed by any of the local sensors ($i = 1,\dots,N$), can be described by the following equation:

$$z_i(t) = H_i(t)x(t) + \xi_i(t), \tag{10}$$

where $H_i(t)$ is a $p(i) \times n$ observation matrix, $p(i) \in \mathbb{N}_+$, and $\xi_i \in \mathbb{R}^{p(i)}$ represents zero-mean Gaussian discrete-time measurement noise with a known covariance matrix

$$R_i(t) \triangleq E\left\{\xi_i(t)\left[\xi_i(t)\right]^{\top}\right\}. \tag{11}$$

Note that the size $p(i)$ of the measurement vector z_i can be different for each local sensor.

3. Data synchronization

In the distributed estimation system considered each local processor ($i = 1,\dots,N$) generates estimation data comprised of local state estimates and their corresponding covariance matrices. The processing nodes can circulate these data asynchronously among themselves.

To perform data fusion, in each data-fusion cycle the i-th local processor synchronizes all data obtained from the other processors to a common time moment. The best choice for such a synchronization moment is the time of the last local measurement. Then the respective measurement can be included in data fusion without any modifications. Such an approach will be exercised in this paper.

Let us consider an example time scale, presented in Fig. 2, for the first processing node of the distributed estimation system of Fig. 1(a).

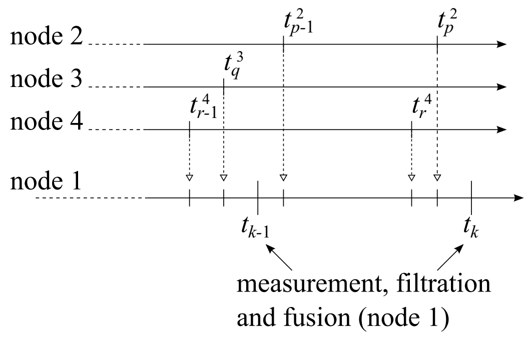

Fig. 2. Example time scale.

The time moments of both data fusion and sensor measurement for this processor are denoted by $t_k, k \in \mathbb{N}$. The time moments of data fusion at nodes 2, 3 and 4 are t^2_p, t^3_q and t^4_r, respectively, with $p,q,r \in \mathbb{N}$. At time t_k the first local processor utilizes data from the second node and from the fourth one attributed to the time moments t^2_p and t^4_r, respectively. Data from the second processor for time t^2_{p-1} are not considered, as newer data (from time t^2_p) are available. On the other hand, no new data from the third node are available (since the previous data fusion at time t_{k-1}).

In the first local processor, for synchronization purposes the estimate $\hat{x}_j(\tau|\tau)$ of the process state gained from the j-th local processor ($j = 2,3,4$) calculated using the data up to time τ can be predicted for time t_k according to the following equation:

$$\hat{x}_j(t_k|\tau) = F(t_k,\tau)\hat{x}_j(\tau|\tau) + u(t_k,\tau). \tag{12}$$

The above equation is based on the c-t model (5). The noise w in (5) is unknown, although the mean of this noise is zero. The corresponding covariance matrix is

$$P_j(t_k|\tau) = F(t_k,\tau)P_j(\tau|\tau)\left[F(t_k,\tau)\right]^{\top} + Q_j(t_k,\tau). \tag{13}$$

The state transition matrix $F(t_k,\tau)$, the input signal $u(t_k,\tau)$ and the covariance matrix $Q(t_k,\tau)$ of the process noise w can be determined according to (6), (7) and (9), respectively. From (9) it is clear that the elements of the matrix Q are in proportion to the time difference between t_k and τ. In a sense, the matrix Q represents a loss of 'information' about the corresponding state estimate $\hat{x}_j$.

With the use of (12) and (13) all the state estimates considered in each data-fusion cycle in any local processor can be synchronized to a common time moment. Such synchronization, executed by each of the processors, allows performing data fusion.

Note, however, that the ample time scale in Fig. 2 is for the first node. Other nodes perform filtration and fusion according to the same algorithm. Therefore in this example case, at time t^2_p the second node used all data from the all other nodes (from the third node from time

t_q^3, from the fourth node from time t_r^4, and even from the first node from time t_{k-1}). So the estimate for time t_p from the second node contains a lot of common (for all the nodes) information. In a distributed estimation system it is not easy to track all data exchanges between nodes and to remove this common information. Therefore, an appropriate data fusion algorithm is used instead, which takes into account unknown cross-correlation between data from different nodes to perform consistent data fusion.

4. Multi-sensor ADE algorithm using CI fusion

The proposed ADE algorithm is based on a standard Kalman filter algorithm (Kalman, 1960; Bar-Shalom and Li, 1993; Blackman and Popoli, 1999), and is an asynchronous extension of the procedure presented by Hall and Llinas (2001, Chapter 12). A general procedural framework for asynchronous data fusion in the first local processors (denoted with the subscript 1) using Kalman filters and data synchronization is described in the following. The other local processors work according to the same algorithm. The complete flowchart of the algorithm is given in Fig. 3.

4.1. ADE algorithm.

Step 1. Calculate the state estimate prediction $\hat{x}_1^0(t_k|t_{k-1})$ and the corresponding covariance matrix $P_1^0(t_k|t_{k-1})$ by utilizing the process model, the results from the previous cycle of the fusion algorithm, and the Kalman prediction equations

$$\begin{aligned}\hat{x}_1^0(t_k|t_{k-1}) = F(t_k, t_{k-1})\hat{x}_1(t_{k-1}|t_{k-1}) \\ + u(t_k, t_{k-1}),\end{aligned} \tag{14}$$

$$\begin{aligned}P_1^0(t_k|t_{k-1}) =& F(t_k, t_{k-1})P_1(t_{k-1}|t_{k-1})\left[F(t_k, t_{k-1})\right]^\top \\ &+ Q(t_k, t_{k-1}).\end{aligned} \tag{15}$$

The upper index 0 means that the results are local and do not include data from other processing nodes.

Step 2. Compute the local estimate by processing the predicted estimate and the local measurement with the use of the following Kalman filter approach:

Compute the measurement prediction

$$\hat{z}_1^0(t_k|t_{k-1}) = H_1(t_k)\hat{x}_1^0(t_k|t_{k-1}) \tag{16}$$

and the covariance matrix of this prediction

$$S_1^0(t_k) = H_1(t_k)P_1^0(t_k|t_{k-1})\left[H_1(t_k)\right]^\top + R_1(t_k), \tag{17}$$

where $R_1(t_k)$ is the covariance matrix (11) of the measurement noise ξ_1.

The gain matrix of the Kalman filter is described as

$$K_1^0(t_k) = P_1^0(t_k|t_{k-1})\left[H_1(t_k)\right]^\top \left[S_1^0(t_k)\right]^{-1}. \tag{18}$$

The Kalman filter innovation is

$$r_1^0(t_k) = z_1(t_k) - \hat{z}_1^0(t_k|t_{k-1}), \tag{19}$$

where $z_1(t_k)$ is a measurement vector, gained from the corresponding sensor, and associated with the time moment t.

The local state estimate for time t is computed as

$$\hat{x}_1^0(t_k|t_k) = \hat{x}_1^0(t_k|t_{k-1}) + K_1^0(t_k)r_1^0(t_k) \tag{20}$$

with the effective covariance matrix expressed by the following Joseph formula:

$$\begin{aligned}P_1^0(t_k|t_k) =& K_1^0(t_k)R_1(t_k)\left[K_1^0(t_k)\right]^\top \\ &+ \left[I - K_1^0(t_k)H_1(t_k)\right]P_1^0(t_k|t_{k-1}) \\ &\times \left[I - K_1^0(t_k)H_1(t_k)\right]^\top,\end{aligned} \tag{21}$$

where I stands for the identity matrix.

The results $\hat{x}_1^0(t_k|t_k)$ and $P_1^0(t_k|t_k)$ are propagated to other processing nodes.

Step 3. Synchronize all the (local) state estimates ($j = 2, \ldots, N$) received from other nodes, according to (12) and (13). The synchronization time is the time for which the predictions $\hat{x}_1^0(t_k|t_{k-1})$ and $P_1^0(t_k|t_{k-1})$ from Step 1 are calculated, i.e., t_k.

Step 4. Perform the fusion of the synchronized estimates from Step 3 with the local predicted estimate from Step 1 using the Covariance Intersection (CI) method[1] (Julier and Uhlmann, 1997; Kowalczuk and Domżalski, 2009; Chen *et al.*, 2002) or any other method of data fusion (Baranski *et al.*, 2011; Bar-Shalom and Li, 1995, Chapter 8), if the CI is computationally too expensive.

The CI is a useful method for data fusion in cases where cross correlation between estimation errors for different nodes is unknown or hard to calculate. Data fusion is then performed according to the following equations (the time indices are omitted here):

$$\begin{aligned}(P_1)^{-1} =& \omega_1\left(P_1^0\right)^{-1} + \omega_2\left(P_2^0\right)^{-1} + \cdots \\ &+ \omega_N\left(P_N^0\right)^{-1},\end{aligned} \tag{22}$$

$$\begin{aligned}\hat{x}_1 =& P_1\Big[\omega_1\left(P_1^0\right)^{-1}\hat{x}_1^0 + \omega_2\left(P_2^0\right)^{-1}\hat{x}_2^0 + \cdots \\ &+ \omega_N\left(P_N^0\right)^{-1}, \hat{x}_N^0\Big]\end{aligned} \tag{23}$$

where $\hat{x}_j^0$ and P_j^0 ($j = 2, \ldots, N$) are the local state estimates and their corresponding covariance matrices

[1]As the cross-correlation between data is unknown.

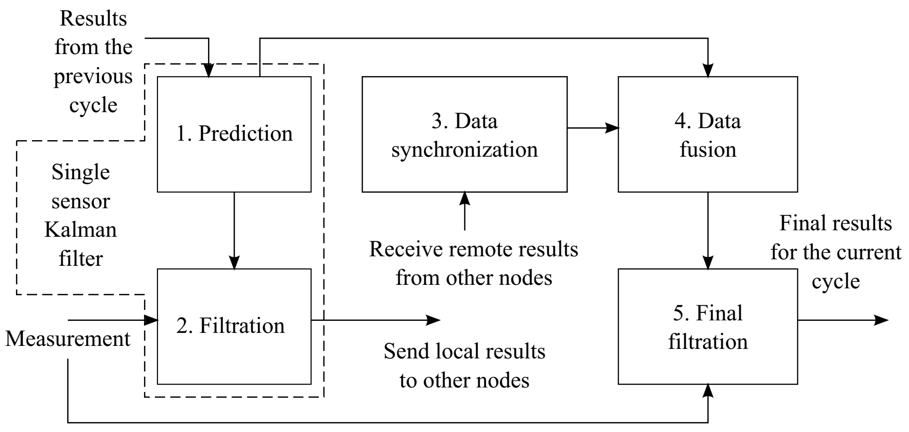
Results from the previous cycle
1. Prediction
Single sensor Kalman filter
2. Filtration
Measurement
3. Data synchronization
Receive remote results from other nodes
Send local results to other nodes
4. Data fusion
Final results for the current cycle
5. Final filtration

Fig. 3. Flowchart of the ADE algorithm.

received from all the nodes, synchronized in Step 3, together with the local predicted state estimate from Step 1 ($j = 1$), $\hat{x}_1$ and P_1 are the resulting (aggregate) state estimate and its corresponding covariance matrix for the considered first node, and $\omega_i \in [0,1]$ ($\sum_{i=1}^{i=N} \omega_i = 1$) are the weighting parameters. In practice, the parameters ω_i are chosen so as to minimize given performance criteria, like the trace or the determinant of P_1. Such optimization must be performed to guarantee that the fusion algorithm is non-divergent. The CI algorithm guarantees that the fused state estimate is consistent.

A classical (and computationally less expensive than CI) algorithm for fusion of data from multiple sources is the following:

$$(P_1)^{-1} = \left(P_1^0\right)^{-1} + \left(P_2^0\right)^{-1} + \cdots + \left(P_N^0\right)^{-1}, \quad (24)$$

$$\hat{x}_1 = P_1 \Big[\left(P_1^0\right)^{-1} \hat{x}_1^0 + \left(P_2^0\right)^{-1} \hat{x}_2^0 + \cdots + \left(P_N^0\right)^{-1} \hat{x}_N^0 \Big]. \quad (25)$$

Note that the above algorithm is similar to the CI method, but with all the weights equal to one. However, by contrast to the CI method, this algorithm does not guarantee consistency in the presence of cross-correlation between data the fused state estimate.

From the above step a new predicted state estimate $\hat{x}_1(t_k|t_{k-1})$ (denoted without the upper index 0) results with the corresponding covariance matrix $P_1(t_k|t_{k-1})$.

Step 5. Perform the ultimate Kalman filtration of the local measurement $z_1(t_k)$ with the fused predicted estimate $\hat{x}_1(t_k|t_{k-1})$ from Step 4 treated as the prior prediction $\hat{x}_1^0(t_k|t_{k-1})$, based on (16)–(21). In such a way the ultimate local state estimate $\hat{x}_1(t_k|t_k)$ and the corresponding covariance matrix $P_1(t_k|t_k)$ are effectively computed. These results are used in the next cycle of the fusion algorithm.

4.2. Remarks on optimality. It is clear that a single Kalman filter used in each node is an optimal estimator for processes described by linear stochastic dynamic models, but only for the subset of measurements available for this node. Moreover, the method of data fusion applied is not optimal since the algorithm fuses the estimates with the use of their covariance matrices solely, i.e., without referring to precise information on cross-correlations between the estimation errors of different nodes. At least, a consistent estimation in terms of the states and their covariances is performed. The consequence of existing cross-correlations is thus reduced to the resulting covariance matrix of the estimate being fused that is cautiously evaluated by the covariance intersection method. The 'caution' effect can be attributed here to the upper bound of all possible resulting covariance matrices. As an effect, the estimation is consistent in terms of a cautious estimation of the error obtained in the CI-based fusion.

5. Simulation example

In this section we show two simulation examples. The first one concerns the problem of state estimation of an Ornstein–Uhlenbeck stochastic process. The second example deals with estimation of two crossing trajectories in the presence of additional false detections (clutter).

5.1. OU process. Let us consider a problem of distributed state estimation of a two-dimensional Ornstein–Uhlenbeck (OU) stochastic process describing the movement of a particle in a Cartesian plane. The OU process can be utilised, for example, as a model of a moving object in air traffic control systems or mobile robotics.

The state equation for velocity v of the OU process is the following (Rogers and Williams, 2000):

$$\begin{aligned} \mathrm{d}v(t) = \begin{bmatrix} \mathrm{d}v_x(t) \\ \mathrm{d}v_y(t) \end{bmatrix} &= \kappa\,(\theta - v(t))\,\mathrm{d}t + \sigma \mathrm{d}w(t) \\ &= \begin{bmatrix} \kappa_{11} & \kappa_{12} \\ \kappa_{21} & \kappa_{22} \end{bmatrix} \left(\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} - \begin{bmatrix} v_x(t) \\ v_y(t) \end{bmatrix} \right) \mathrm{d}t \\ &\quad + \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix} \begin{bmatrix} \mathrm{d}w_x(t) \\ \mathrm{d}w_y(t) \end{bmatrix} \end{aligned} \tag{26}$$

and the equation for the position p is simply

$$\mathrm{d}p(t) = \begin{bmatrix} \mathrm{d}p_x(t) \\ \mathrm{d}p_y(t) \end{bmatrix} = \begin{bmatrix} v_x(t) \\ v_y(t) \end{bmatrix} \mathrm{d}t = v(t)\,\mathrm{d}t, \tag{27}$$

where $\mathrm{d}w$ is an infinitesimal increment of the two-dimensional Brownian motion, the matrix κ describes the rate of velocity mean reversion, the vector θ is the long-term mean of the velocity, and the diagonal matrix σ describes the average magnitude of random velocity fluctuations.

Let us assume the following parameters of the process:

$$\kappa = \begin{bmatrix} 0.05 & 0.02 \\ -0.04 & 0.1 \end{bmatrix}, \quad \theta = \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \quad \sigma = \begin{bmatrix} 0.2 & 0 \\ 0 & 0.2 \end{bmatrix} \tag{28}$$

and the initial state

$$\begin{bmatrix} p_x(0) & p_y(0) & v_x(0) & v_y(0) \end{bmatrix}^\top = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}^\top. \tag{29}$$

Sensors are assumed to measure the position of the process, solely. Therefore, the common observation matrix for all sensors is

$$H_1 = H_2 = H_3 = H_4 = H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}. \tag{30}$$

Now let us consider two different configurations of the sensors. In the first configuration, the system shown in Fig. 1(a) is assumed, which has four local nodes and their corresponding sensors ($i = 1, 2, 3, 4$) and represents a complete connection graph between the nodes, whereas the connection graph applied in a second configuration will not be complete (as will be described later).

In the experiments performed in the first configuration all the sensors adopt the following sampling scheme. In each time interval $(t, t + 1)$, $t = 0, 1, 2, \ldots 59$ (the final time was 60 s), each sensor performed one measurement at random. Sample values for the sampling times for the first sensor could be $(0.396, 1.798, 2.134, \ldots)$, and for the second sensor $(0.743, 1.347, 2.014, \ldots)$, etc. An average sampling rate for all the sensors was one measurement per second. The real time of a measurement, once determined, was used for data synchronisation.

What differentiated the sensors in the first configuration was their accuracy described by the following (time constant) measurement covariance matrices:

$$R_1 = \begin{bmatrix} 5 & 0 \\ 0 & 0.3 \end{bmatrix}, \qquad R_2 = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix},$$
$$R_3 = \begin{bmatrix} 2 & 0 \\ 0 & 2 \end{bmatrix}, \qquad R_4 = \begin{bmatrix} 0.5 & 0 \\ 0 & 4 \end{bmatrix}. \tag{31}$$

Clearly, the measurement accuracy for position x is the worst for sensor 1, slightly better for sensor 2, better for sensor 3, and the best for sensor 4. This effect is reversed for position y, with sensor 1 having the best accuracy and sensor 4 being the worst in terms of accuracy.

The announced second configuration of the system is comprised of the same nodes as in the first configuration. This time, however, the connection graph for the ADE system is not complete, as presented in Fig. 1(b). For example, note that the first node (the worst at measuring position x) and the fourth node (the best at measuring position x) are not directly connected; instead, there are two connecting (intermediate) nodes between them.

For a proper exposition of the performance of asynchronous distributed estimation, their results should be compared with those of estimation using both the Kalman filter based on data from a single sensor and the optimal (centralised) estimator utilising all raw measurements from all the nodes.

We have defined the estimation error as the average absolute value (over 2000 trajectories) of the difference between the true value computed by the trajectory simulator and the value estimated by the discussed algorithms.

Estimation errors for the position (p_x and p_y) and velocity (v_x and v_y) for the first and the fourth nodes working in the first configuration are presented in Figs. 4 and 5, respectively. The results for these two nodes are the two extreme cases of sensor accuracy, with the results for the second and the third node inbetween. Therefore, the results for the second and the third node are not presented.

The sensor corresponding to the first node is the best at measuring the position in y and the worst at measuring the position in x as compared to the sensors corresponding to the other nodes. Therefore, the estimates of the position in x and the velocity in x for the first node are significantly improved by using data from the other nodes (Figs. 4(a) and (c), respectively), whereas there is almost no improvement in the estimates of the position in y and the velocity in y when using data from the other nodes (Figs. 4(b) and (d), respectively), as compared to the Kalman filter. The results are reversed for the fourth node. There is almost no improvement of estimates of the position in x and the velocity in x (Figs. 5(a) and (c)), whereas a significant improvement can be observed in the case of estimating the position in y and the velocity in y (Figs. 5(b) and (d)).

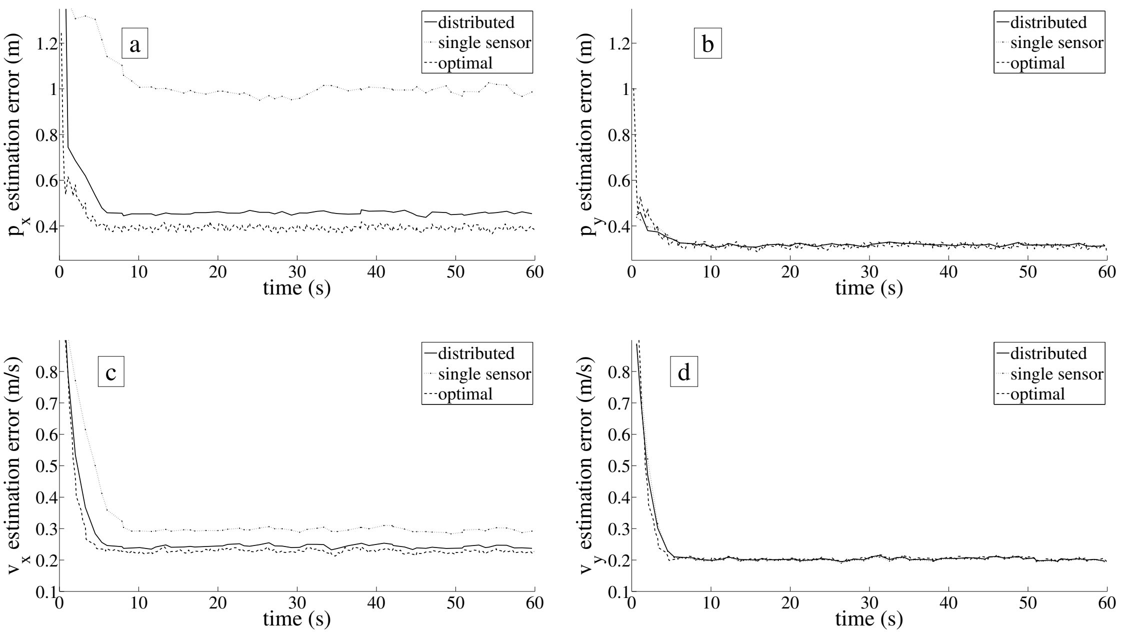

Fig. 4. Estimation errors for the first node: position p_x (a), position p_y (b), velocity v_x (c), velocity v_y (d).

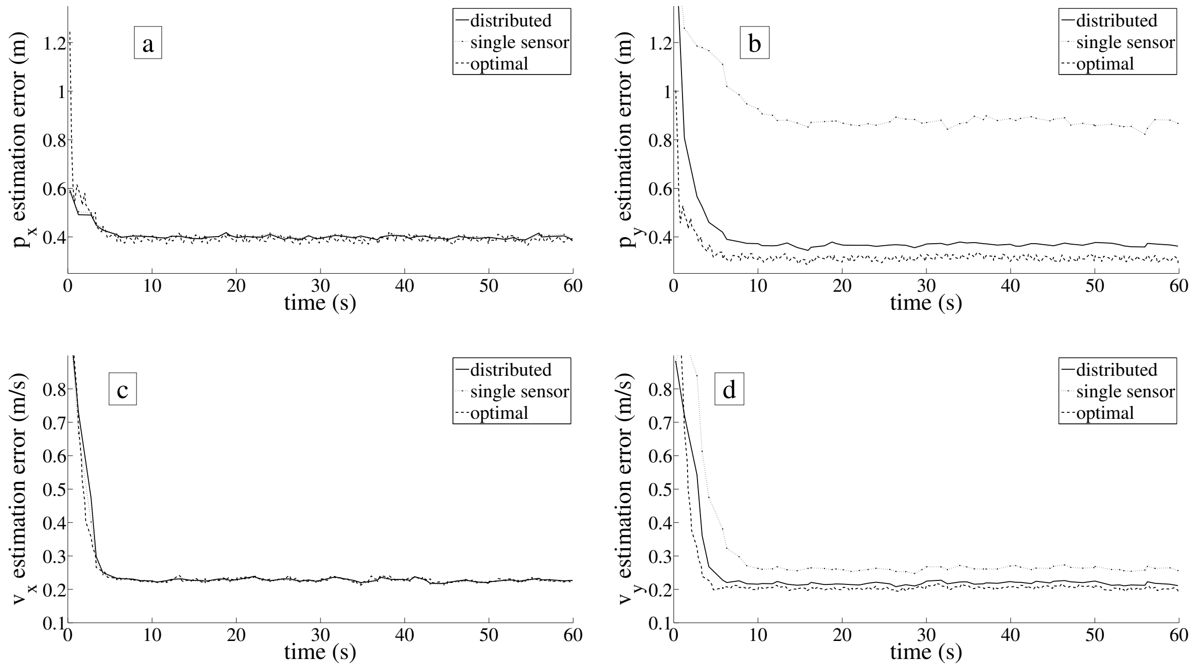

Fig. 5. Estimation errors for the fourth node: position p_x (a), position p_y (b), velocity v_x (c), velocity v_y (d).

Note that the improved state estimates of the first node for the coordinate x (the continuous-line estimates in Figs. 4(a) and (c)) are still worse than their counterparts corresponding to the fourth sensor (Figs. 5(a) and (c)). Thus the information exchange between the nodes is not perfect and the errors of the first node for the coordinate x are larger than those of the optimal estimator, whereas the errors of the fourth node in the case of the coordinate x are equal to those of the optimal estimator.

The results of estimating the position x for the first node working within the system of the incomplete connection graph, compared with the results discussed above for the same node working in the system of the complete connection graph, are presented in Fig. 6.

Since the first node has no direct connection with the fourth node, accurate estimates of position x gained by the fourth node are transmitted to the first node with a certain delay. Clearly, this delay leads to a degradation in the estimation performance of the first node, as the delay makes the (good) estimates of the state invalid for the dynamic process considered. To confirm this conclusion, an additional experiment was performed. Namely, estimation of a static process (with constant parameters) perturbed solely by measurement noise. The obtained results are presented in Fig. 7. For such a static process with a constant state, the transmission delay does not invalidate the state estimates being exchanged. Therefore, the estimation error for the first node in the system with the incomplete graph converges to the error of the same node in the system with the complete graph.

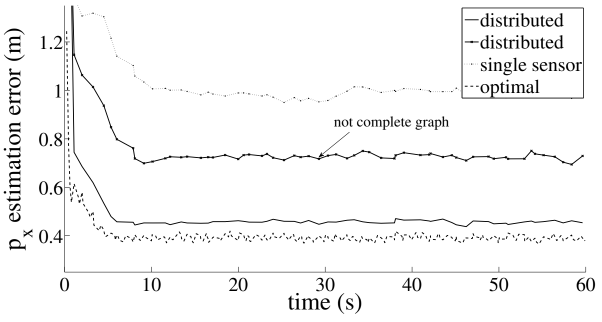

Fig. 6. Position estimation errors for the first node in the second configuration.

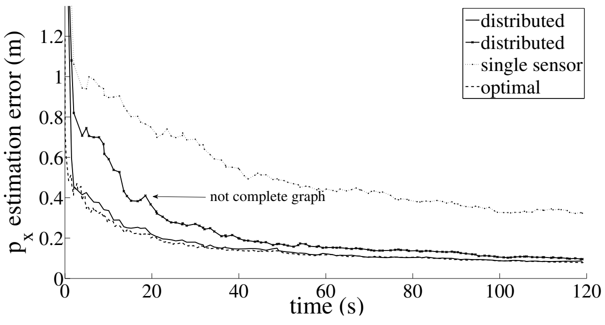

Fig. 7. Position estimation errors for the first node: estimation of a static value.

More extensive results for the above examples and the results for another configuration of the local processors are presented in the work of Kowalczuk and Domżalski (2012b).

5.2. Two trajectories.

In this section we consider the problem of estimating crossing trajectories of two moving objects in a 2D Cartesian space. In this case, instead of a Kalman filter, the estimation algorithm in a local processor is based on a joint probabilistic data association filter (Fotmann *et al.*, 1980; 1983; Fitzgerald, 1985; Bar-Shalom and Li, 1995; Chen and Tugnait, 2001). The JPDA filter makes an efficient method for estimating trajectories of multiple objects moving in a common space in the presence of undesired false detections.

In the performed experiments, both objects were described by two-dimensional OU processes. The parameters for the first object were

$$\kappa_1 = \begin{bmatrix} 0.05 & 0.02 \\ -0.04 & 0.1 \end{bmatrix}, \quad \theta_1 = \begin{bmatrix} 2 \\ -1 \end{bmatrix}, \quad \sigma_1 = \begin{bmatrix} 0.01 & 0 \\ 0 & 0.01 \end{bmatrix}, \tag{32}$$

with the following initial state:

$$\begin{aligned} X_1(0) &= \begin{bmatrix} p_x^1(0) & p_y^1(0) & v_x^1(0) & v_y^1(0) \end{bmatrix}^\top \\ &= \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}^\top . \end{aligned} \tag{33}$$

For the second object, the model matrices were the following:

$$\kappa_2 = \kappa_1, \quad \theta_2 = \begin{bmatrix} -1 \\ -2 \end{bmatrix}, \quad \sigma_2 = \sigma_1, \tag{34}$$

and the initial state was

$$\begin{aligned} X_2(0) &= \begin{bmatrix} p_x^2(0) & p_y^2(0) & v_x^2(0) & v_y^2(0) \end{bmatrix}^\top \\ &= \begin{bmatrix} 32 & 5 & -1 & 1 \end{bmatrix}^\top . \end{aligned} \tag{35}$$

The ADE system from Fig. 1(a) was utilized, with the same sampling scheme as the one described in the example from Section 5.1. The only difference was that (for this case) all the measurement noise covariance matrices were the same and equal to

$$R_1 = R_2 = R_3 = R_4 = \begin{bmatrix} 0.8 & 0 \\ 0 & 0.8 \end{bmatrix}. \tag{36}$$

The JPAD filter applied allows us to estimate trajectories of moving objects in the presence of undesired false detection[2]. However, for time $t < 5$, false detections were not simulated in order to allow the JPDA filters to perform robust track initialization. For time $t \geq 5$, for each node, the number of false detections in each cycle was assumed to have the Poisson probability distribution described by the following probability mass function:

$$P(m) = \frac{(\lambda \times \text{Vol})^m e^{-\lambda \times \text{Vol}}}{m!}, \tag{37}$$

where λ is a sensor-dependent density of false detections, chosen for this example as $\lambda = 0.1$, and 'Vol' is the surface of the space observed by the sensors.

[2]By broad definition, a false detection is anything that is not a valid object, e.g., noise originating detections or detections from stationary structures in the observed space.

Example results of tracking the two trajectories, for time $t \in [0, 20]$, are presented in Fig. 8. The simulated trajectories are represented by dashed lines, the objects originated measurements have the symbol $\times$, and the results of estimation are denoted with the token $\circ$ and connected with the use of straight lines. Additional false detections simulated in the last cycle of the algorithm are represented by the symbol $+$.

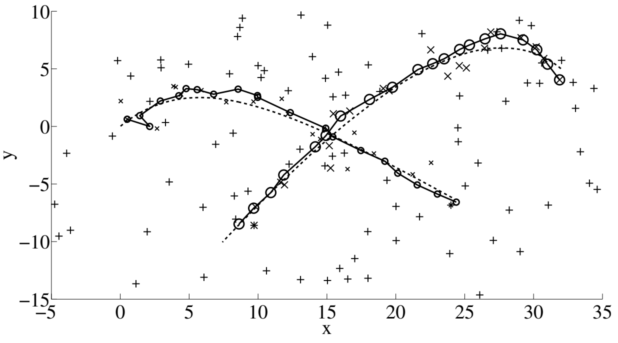

Fig. 8. Example results of the tracking of two trajectories using the JPDA filter.

In the discussed case the observed space was defined as a rectangle with its lower left vertex at $(-5, -15)$ and its upper right vertex at $(35, 10)$. Thus the surface of the observed space was Vol $= 1000$, and in each cycle, on average, $\lambda \times \text{Vol} = 0.1 \times 1000 = 100$ false detections were created with a uniform distribution over the observed space.

The results of estimation of the position p_x and p_y and the velocity v_x i v_y for the first trajectory in the first node are presented in Fig. 9. As the results for the other (identical) nodes were similar, they are not attached here.

As in the case of estimating a single trajectory (example in Section 5.1), the results for the distributed system are slightly worse than those for the optimal (central) processing, and are much better than those for the single sensor systems.

For time $t \in [0, 5]$ the estimation errors decline since, as mentioned earlier, no additional false detections were simulated. The estimation errors for all algorithms have their maximum for time $t \in [10, 12]$, that is, during the time of crossing the two trajectories. For time $t > 15$ the objects move apart and the errors suitably decline again.

6. Conclusions

In this paper we have discussed a new framework for asynchronous distributed estimation based on continuous-time stochastic models. In general, we have assumed that nodes of such an estimation system are not synchronized and work independently of each other. From a systemic viewpoint, very important is that one can easily add new nodes to such systems and in the case of a failure in any node the rest of the nodes can continue to operate independently. In this respect the proposed ADE system is thus flexible and robust.

From the presented results it is clear that the performance of the nodes improves if they exchange information with each other. The difference between the ADE system and a single sensor Kalman filter is even more apparent if one node sensor performs inaccurate measurements (i.e., its measurement noise variance is large) compared with the other ones. In such cases, data from more accurate nodes positively support the work of this node, i.e, improve its local estimates.

Compared with the optimal estimator, the results for the presented ADE algorithm are slightly worse. The basic difficulty lies in the fact that each node exchanges data with other nodes completely asynchronously, and that in real distributed systems getting necessary cross-correlations is practically impossible. In fact, the asynchronous interaction between the nodes of the ADE system stymies analytical estimation of cross-correlations between the estimation errors.

In spite of that, the problem of optimality of ADE has also been addressed. Note that a single Kalman filter is an optimal estimator for plants described by linear stochastic dynamic models, but solely for the subset of measurements available at this node (a centralized globally optimal filter uses all measurements). The method of data fusion applied cannot be fully optimal, because the algorithm fuses the estimates solely with the use of their covariance matrices (and without referring to precise information on cross-correlations between the estimation errors of different nodes). Thus the ADE estimation is consistent in terms of the states and their covariances. This also means that the effect of existing cross-correlations is reduced to the resulting covariance matrix of the fused estimate that is cautiously evaluated with the use of the covariance intersection method.

The case of systems described by their connection graphs being not complete has also been considered. In such systems, the lack of connections between certain nodes introduces some transmission delay. This delay leads to a degradation in the estimation performance, since the delay invalidates the exchanged estimates of the state in the case of dynamic objects. For static ones the estimation error in the ADE systems with incomplete graphs converges to an error corresponding to the system configuration with a complete connection graph. Such an effect of data degradation can be more evident for systems comprised of a large number of nodes. Therefore, great care should be taken in selecting the system configuration with incomplete connections. Moreover, a general conclusion can be drawn for such cases that an acceptable delay introduced by the ADE system should be comparable with the time constant (dynamics) of the observed plant.

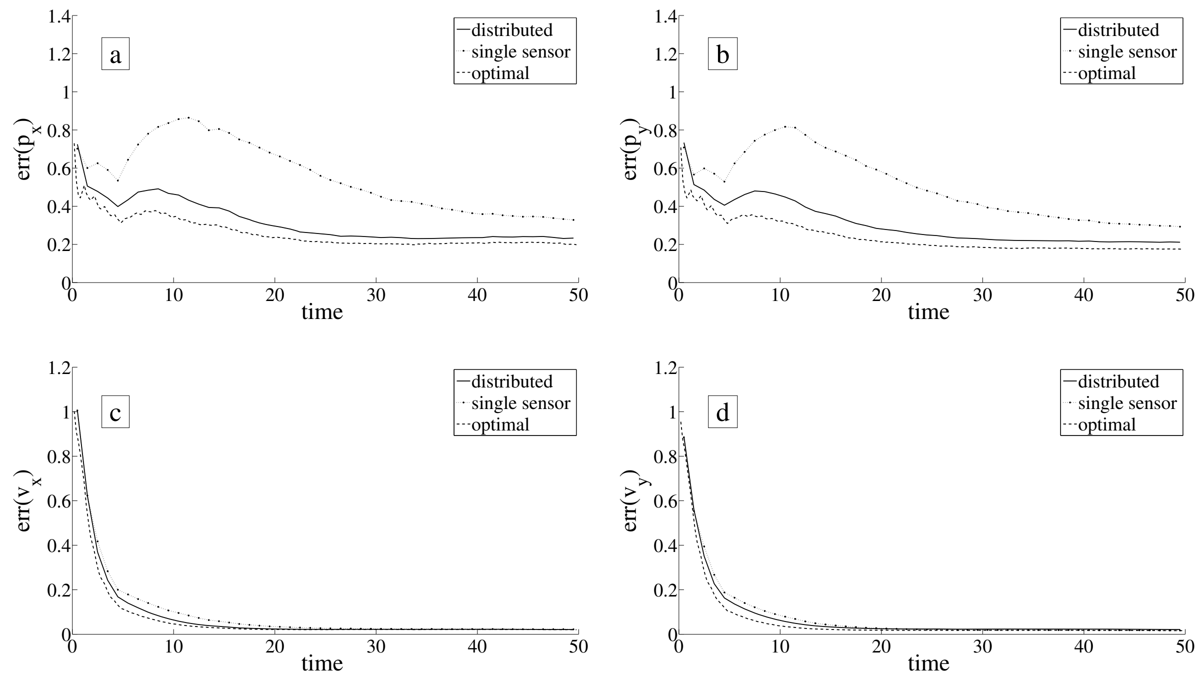

Fig. 9. Estimation errors for the first trajectory in the first node (JDPA filter): position p_x (a), position p_y (b), velocity v_x (c), velocity v_y (d).

Note that the above conclusions are also valid for ADE systems utilizing other estimation algorithms, e.g., JPDA filters.

It appears that in future studies it can be worth considering the issues of data synchronization, based on a more general stochastic process model, implementation of the presented estimation algorithm on an embedded computer used for mobile robotics, as well as evaluating certain nonlinear problems which may happen in real (air traffic control) multi-sensor systems.

References

Baranski, P., Polanczyk, M. and Strumillo, P. (2011). Fusion of data from inertial sensors, raster maps and GPS for estimation of pedestrian geographic location in urban terrain, *Metrology and Measurement Systems* **18**(1): 145–158.

Bar-Shalom, Y. and Li, X.R. (1993). *Estimation and Tracking: Principles, Techniques, and Software*, Artech House, Boston, MA.

Bar-Shalom, Y. and Li, X.R. (1995). *Multitarget-Multisensor Tracking: Principles and Techniques*, YBS Publishing, Storrs, CT.

Bilenne, O. (2004). Fault detection by desynchronized Kalman filtering: Introduction to robust estimation, *in* P. Svensson and J. Schubert (Eds.), *Proceedings of the Seventh International Conference on Information Fusion*, Vol. I, International Society of Information Fusion, Mountain View, CA, pp. 99–106.

Blackman, S. and Popoli, R. (1999). *Design and Analysis of Modern Tracking Systems*, Artech House, Boston, MA.

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

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

Chen, B. and Tugnait, J. (2001). Tracking of multiple maneuvering targets in clutter using IMM/JPDA filtering and fixed-lag smoothing, *Automatica* **37**(2): 239–249.

Chen, L., Arambel, P.O. and Mehra, R.K. (2002). Estimation under unknown correlation: Covariance intersection revisited, *IEEE Transactions on Automatic Control* **AC-47**(11): 1879–1882.

Chen, Y.-L. and Chen, B.-S. (1995). Optimal reconstruction of ARMA signals with decimated samples under corrupting noise by use of multirate Kalman filter, *Circuits, Systems, Signal Processing* **14**(6): 771–786.

Del Favero, S. and Zampieri, S. (2009). Distributed estimation through randomized gossip Kalman filter, *Proceedings of the 48th IEEE Conference on Decision and Control*.

Fitzgerald, R. (1985). Track biases and coalescence with probabilistic data association, *IEEE Transactions on Aerospace and Electronic Systems* **21**(6): 822–825.

Fortmann, T., Bar-Shalom, Y. and Scheffe, M. (1980). Multi-target tracking using joint probabilistic data association, *Proceedings of 1980 IEEE Conference on

Decision and Control, Albuquerque, NM, USA, pp. 807–812.

Fortmann, T., Bar-Shalom, Y. and Scheffe, M. (1983). Sonar tracking of multiple targets using joint probabilistic data association, *IEEE Journal of Oceanic Engineering* **8**(3): 173–184.

Hall, D.L. and Llinas, J. (1997). An introduction to multisensor data fusion, *Proceedings of the IEEE* **85**(1): 6–23.

Hall, D. L. and Llinas, J. (2001). *Handbook of Multisensor Data Fusion*, CRC, Boca Raton, FL.

Julier, S. and Uhlmann, J. (1997). A non-divergent estimation algorithm in the presence of unknown correlations, *Proceedings of the American Control Conference, Albuquerque, NM, USA*, pp. 2369–2373.

Kalman, R. (1960). A new approach to linear filtering and prediction problems, *Transactions of the ASME, Journal of Basic Engineering* **82**(1): 34–45.

Karatzas, I. and Shreve, S.E. (1991). *Brownian Motion and Stochastic Calculus*, Springer, New York, NY.

Kowalczuk, Z. and Domżalski, M. (2009). Asynchronous distributed state estimation based on covariance intersection, *System Science* **35**(1): 23–30.

Kowalczuk, Z. and Domżalski, M. (2012a). Optimal asynchronous estimation of 2D Gaussian–Markov processes, *International Journal of Systems Science* **43**(8): 1431–1440.

Kowalczuk, Z. and Domżalski, M. (2012b). Asynchronous distributed state estimation based on a continuous-time stochastic model, *International Journal of Adaptive Control and Signal Processing* **26**(5): 384–399.

Kuchler, R. and Therrien, C. (2003). Optimal filtering with multirate observations, *Proceedings of the 37th Asilomar Conference on Signals, Systems, and Computers, Pacific Grove, CA, USA*, pp. 1208–1212.

Liggins, M., Chong, C., Kadar, I., Alford, M., Vinnicola, V. and Thomopoulos, S. (1997). Distributed fusion architectures and algorithms for target tracking, *Proceedings of the IEEE* **85**(1): 95–107.

Oksendal, B. (2003). *Stochastic Differential Equations: An Introduction with Applications*, Springer-Verlag, Berlin.

Olfati-Saber, R. (2007). Distributed Kalman filtering for sensor networks, *Proceedings of the 46th IEEE Conference on Decision and Control, New Orleans, LA, USA*, pp. 5492–5498.

Olfati-Saber, R. and Shamma, J. (2005). Consensus filters for sensor networks and distributed sensor fusion, *Proceedings of the 44th IEEE Conference on Decision and Control/2005 European Control Conference (CDC-ECC 05), Seville, Spain*, pp. 6698–6703.

Patan, M. (2012). Distributed scheduling of sensor networks for identification of spatio-temporal processes, *International Journal of Applied Mathematics and Computer Science* **22**(2): 299–311, DOI: 10.2478/v10006-012-0022-9.

Rao, B., Durrant-Whyte, H. and Sheen, J. (1993). A fully decentralized multi-sensor system for tracking and surveillance, *International Journal of Robotics Research* **12**(1): 20–44.

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

Rogers, L. and Williams, D. (2000). *Diffusion, Markov Processes and Martingales, Vol. 2: Itô Calculus*, Cambridge University Press, Cambridge.

Speranzon, A., Fischione, C. and Johansson, K. (2006). Distributed and collaborative estimation over wireless sensor networks, *Proceedings of the 45th IEEE Conference on Decision and Control, San Diego, CA, USA*, pp. 1025–1030.

Uciński, D. (2012). Sensor network scheduling for identification of spatially distributed processes, *International Journal of Applied Mathematics and Computer Science* **22**(1): 25–40, DOI: 10.2478/v10006-012-0002-0.

Xiao, L., Boyd, S. and Lall, S. (2005). A scheme for robust distributed sensor fusion based on average consensus, *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, Los Angeles, CA, USA*, pp. 63–70.

Zhang, H., Basin, M. and Skliar, M. (2007). Itô–Volterra optimal state estimation with continuous, multirate, randomly sampled, and delayed measurements, *IEEE Transactions on Automatic Control* **52**(3): 401–416.

Zdzisław Kowalczuk, Prof., D.Sc., Ph.D., M.Sc.E.E. (2003, 1993, 1986, 1978). Since 1978 he has been with the Faculty of Electronics, Telecommunications and Computer Science at the Gdańsk University of Technology, where he is a full professor of automatic control and the chair of the Department of Decision Systems. He held visiting appointments at the University of Oulu (1985), Australian National University (1987), Technische Hochschule Darmstadt (1989), and George Mason University (1990–1991). His main interests include adaptive and predictive control, system identification, failure detection, signal processing, artificial intelligence, control engineering and computer science. He has authored and co-authored about 15 books and 45 book chapters, about 85 journal papers and 200 conference publications. He is a recipient of the 1990 and 2003 Research Excellence Awards of the Polish National Education Ministry, and a 1999 Polish National Science Foundation Award in automatic control.

Mariusz Domżalski received an M.Sc. (2002) and a Ph.D. (2012, with honours) from the Gdańsk University of Technology in the domain of automatic control and robotics. His research interests are in continuous time stochastic processes and estimation theory. His current research activities are focused on distributed estimation based on data from multiple sources.

Received: 26 January 2012
Revised: 6 June 2012