Automatica 47 (2011) 1399–1408

Contents lists available at ScienceDirect

Automatica

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

Multisensor fusion for linear control systems with asynchronous, Out-Of-Sequence and erroneous data[☆]

Eva Besada-Portas[a,*], Jose A. Lopez-Orozco[a], Juan Besada[b], Jesus M. de la Cruz[a]

[a] *Departamento de Arquitectura de Computadores y Automática, Universidad Complutense de Madrid, 28040 Madrid, Spain*

[b] *Departamento de Señales, Sistemas y Radiocomunicaciones, Universidad Politécnica de Madrid, 28040 Madrid, Spain*

ARTICLE INFO

Article history:
Received 22 July 2009
Received in revised form
23 December 2010
Accepted 23 January 2011
Available online 17 March 2011

Keywords:
Estimation theory
Information and sensor fusion
Systems with time-delays
Sensor networks
Kalman filtering
Out-of-sequence data

ABSTRACT

This paper presents a set of new centralized algorithms for estimating the state of linear dynamic Multiple-Input Multiple-Output (MIMO) control systems with asynchronous, non-systematically delayed and corrupted measurements provided by a set of sensors. The delays, which make the data available Out-Of-Sequence (OOS), appear when using physically distributed sensors, communication networks and pre-processing algorithms. The potentially corrupted measurements can be generated by malfunctioning sensors or communication errors. Our algorithms, designed to work with real-time control systems, handle these problems with a streamlined memory and computational efficient reorganization of the basic operations of the Kalman and Information Filters (KF & IF). The two versions designed to deal only with valid measurements are optimal solutions of the OOS problem, while the other two remaining are suboptimal algorithms able to handle corrupted data.

1. Introduction

The state of a complex control system is estimated by its fusion center with the data provided by its sensors. The time and order of arrival of the information at the fusion center depends on many factors, such as the physical distribution of the sensors and the communication network used to send the data. A difficult scenario occurs when the delays and the sequence of arrival of the information are not fixed, constituting the named Out-Of-Sequence Problem (OOSP) (Bar-Shalom, 2002). Another important problem happens when some data are provided by malfunctioning sensors or corrupted during the communication, and these behaviors are not modeled in the estimation algorithms. The fusion system is then in charge of assessing the validity of the information and deciding how to treat the erroneous data (Hall, 1992). Finally, both problems are usually aggravated in networked real-time control systems because the solution adopted to tackle them can affect the stability of its feedback loops (Hespanha, Naghhtabrizi, & Xu, 2007; Sinopoli et al., 2004).

In the case of sequential fusion algorithms, such as the KF and IF (Mutambara, 1998), there are three naïve solutions to deal with the OOSP. The first, rejecting the delayed information, is only appropriated with spurious delayed measurements because it increases the uncertainty and reduces the reliability of the control system (Hespanha et al., 2007; Sinopoli et al., 2004). The second, buffering all the data related with an instant before estimating its state (Lopez-Orozco, de la Cruz, Besada, & Rupiezed, 2000), is not valid for control systems where the response is needed before all the data are available. Finally, the third consists of storing the estimates of the state, the control signals, and the sensor data for all the time instances; rolling back to the time-stamp associated with the measurement which has just arrived, and re-starting the fusion process from that measurement (Kosaka, Meng, & Kak, 1993). This last solution lets the fusion center obtain the same results as if it had received the data without delays. However, it increases the memory needs of the fusion center and its computational overload introduces delays that can affect the controller. The new OOS versions of some estimators, such as Anxi, Diannong, Weidong, and Zhen (2005), Bar-Shalom (2002), Bar-Shalom, Mallick, Chen, and Washburn (2004), Challa, Evans, Wang, and Leggy (2002), Feng, Ge, and Wen (2008), Hilton, Martin, and Blair (1993), Ito, Tsujimichi, and Kosuge (1998), Lanzkron and Bar-Shalom (2004), Lu, Zhang, Wang, and Teo (2005), Mallick, Coraluppi, and Carthel (2001), Matveev and Savkin (2003), Nettleton and Durrant-Whyte

☆ This work is supported by the Community of Madrid under Project CAM-S-0505/DPI-0391 and by the Spanish Ministry of Science and Technology under Project DPI2009-14552-C02-01 and Project TECT2008-06732-C02-101. The work of E. Besada-Portas was also supported by the Spanish Postdoctoral Grant EX-2007-0915 associated with the Prince of Asturias Endowed Chair of the University of New Mexico, USA. This paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form by Associate Editor Giuseppe De Nicolao under the direction of Editor Ian R. Petersen.

* Corresponding author. Tel.: +34 913944740.
E-mail addresses: evabes@dacya.ucm.es (E. Besada-Portas), jalo@dacya.ucm.es (J.A. Lopez-Orozco), besada@grpss.ssr.upm.es (J. Besada), jmcruz@fis.ucm.es (J.M. de la Cruz).

0005-1098/$ – see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.automatica.2011.02.030

(2001), Rheaume and Benaskeur (2008), Shen, Zhu, Song, and Luo (2009), WenHui, Lin, GuoHai, and AnXi (2006), Zhang, Li, and Chen (2002), Zhang, Li, and Chen (2003), Zhang, Xie, Zhang, and Soh (2004) and Zhang, Li, and Zhu (2005) for the KF, reduce these delays and memory needs.

The erroneous data problem can be tackled by including a validation step for testing if the measurements are coherent with the state of the system and rejecting them when they are not (Hall, 1992). When the OOSP is also present, the two problems interact. On the one hand, a validation test dependent on the estimate of the state that only considers the measurements available so far is influenced by their order of arrival. On the other hand, the estimates will depend on the data which have not been rejected because they successfully pass the validation test.

This paper presents a set of simple memory and computational efficient algorithms designed to estimate the state of MIMO linear control systems with additive non-correlated Gaussian noise in the transition and measurement models with all the valid available data received up to this point with random delays. They extend the range of applicability of our first OOS algorithm, named IFAsyn (IF for Asynchronous data) in Besada-Portas, Lopez-Orozco, and de la Cruz (2007) and Besada-Portas, Lopez-Orozco, Besada, and de la Cruz (2009) and hereafter IFAsyn-I (IFAsyn-version I), to systems that work without prior knowledge of the measurement time-stamps, assimilate multiple measurements with different time-stamps in a single update step, and incorporate a validation step to detect corrupted data. When the validation step is disabled, they find the same optimal solution as the KF. When the validation step is enabled, their results and the KF ones can differ, although our experiments show that the differences are negligible. The new algorithms, hereafter IFAsyn-(II, III, IV, V), also reduce the memory and computational needs of IFAsyn-I.

This paper also includes a comprehensive comparison of our OOS algorithms with many others. In short, all versions of IFAsyn are computationally efficient, simple to implement, and general in scope because they already: (1) include the control signal and (2) consider the multisensor case. Besides, IFAsyn-(II, III, IV, V) work without prior knowledge of the data time-stamps, IFAsyn-(III, V) assimilate multiple measurements in a single iteration, and IFAsyn-(IV, V) include a validation step.

The paper is organized as follows: Section 2 introduces some background, Section 3 describes our algorithms, Section 4 compares them with other algorithms, Section 5 analyzes the influence of the validation step in the results, and finally, Section 6 presents the conclusions.

2. Background

2.1. Problem statement

A discrete MIMO linear control system with additive Gaussian noise and S sensors is modeled[1] by Eq. (1), where $\boldsymbol{x}_t$ is the state of the system at time t; $\boldsymbol{z}_{s,t}$ the measurement of sensor s at time t; $\boldsymbol{u}_{t,t_P}$ the control signal applied from the previous time step t_P to the current t; $\boldsymbol{F}_{t,t_P}$ and $\boldsymbol{H}_{s,t}$ the transition and measurement matrices, and $\boldsymbol{\upsilon}_{t,t_P}$ and $\boldsymbol{\nu}_{s,t}$ random Gaussian variables with zero mean and covariances $\boldsymbol{Q}_{t,t_P}$ and $\boldsymbol{R}_{s,t}$.

$$\begin{aligned} &\boldsymbol{x}_t = \boldsymbol{F}_{t,t_P}\boldsymbol{x}_{t_P} + \boldsymbol{u}_{t,t_P} + \boldsymbol{\upsilon}_{t,t_P} \\ &\boldsymbol{z}_{s,t} = \boldsymbol{H}_{s,t}\boldsymbol{x}_t + \boldsymbol{\nu}_{s,t} \quad \text{with } s = 1:S. \end{aligned} \tag{1}$$

The objective of the fusion algorithm is to estimate the current system state and covariance $(\hat{\boldsymbol{x}}_{t|t}, \boldsymbol{P}_{t|t})$ given its original values $(\boldsymbol{x}_{0|0}, \boldsymbol{P}_{0|0})$, the model parameters and control signals $\{\boldsymbol{F}_{k,k_P}, \boldsymbol{Q}_{k,k_P}, \boldsymbol{H}_{s,k}, \boldsymbol{R}_{s,k}, \boldsymbol{u}_{k,k_P}\}$, and the data $\{\boldsymbol{\xi}_{s,k,a} = \boldsymbol{z}_{s,k} | a \geq k, a \leq t\}$ measured by sensor s at time k, which have arrived at the fusion center at time a $(a \geq k)$, and which is already available $(a \leq t)$. In addition, the algorithm is also responsible for detecting and rejecting erroneous data produced by failures not modeled in the sensor covariance matrices.

2.2. Estimating the state with non-delayed data ($\boldsymbol{\xi}_{s,t,t}$)

When the measurements are available without delays $(\boldsymbol{\xi}_{s,t,t})$, $(\hat{\boldsymbol{x}}_{t|t}, \boldsymbol{P}_{t|t})$ can be obtained by sequentially using the prediction and update steps of the KF (Mutambara, 1998). An equivalent approach, with the same two steps, is the IF (Mutambara, 1998). They operate in two different spaces, KF in the state space $(\hat{\boldsymbol{x}}_{t|t}, \boldsymbol{P}_{t|t})$ and IF in the information space $(\hat{\boldsymbol{y}}_{t|t}, \boldsymbol{Y}_{t|t})$, that are related by the state projection operation $\{\hat{\boldsymbol{y}}_{j|l} = \boldsymbol{P}_{j|l}^{-1}\hat{\boldsymbol{x}}_{j|l}, \boldsymbol{Y}_{j|l} = \boldsymbol{P}_{j|l}^{-1}\}(\perp_S)$. In each iteration they only need their previous time t_P space variables $(\hat{\boldsymbol{x}}_{t_P|t_P}, \boldsymbol{P}_{t_P|t_P}$ or $\hat{\boldsymbol{y}}_{t_P|t_P}, \boldsymbol{Y}_{t_P|t_P})$ and the current time t parameters and data $(\boldsymbol{F}_{t,t_P}, \boldsymbol{Q}_{t,t_P}, \boldsymbol{H}_{s,t}, \boldsymbol{R}_{s,t}, \boldsymbol{u}_{t,t_P}, \boldsymbol{\xi}_{s,t,t})$. Further, as the prediction (Eq. (2)) is simpler in the KF and the update of multiple measurements (Eq. (3)) is easier in the IF (Mutambara, 1998), the estimation problem can be solved by combining KF predictions, IF updates and state projections. Finally, the IF update can be divided in a projection of the measurement into the information space (3)$(\perp_M)$, the accumulation of all the projected measurements (3)$(+)$, and the assimilation of the accumulated data with the previous information (3)(A).

$$\left.\begin{aligned} \hat{\boldsymbol{x}}_{t|t_P} &= \boldsymbol{F}_{t,t_P}\hat{\boldsymbol{x}}_{t_P|t_P} + \boldsymbol{u}_{t,t_P} \\ \boldsymbol{P}_{t|t_P} &= \boldsymbol{F}_{t,t_P}\boldsymbol{P}_{t_P|t_P}\boldsymbol{F}_{t,t_P}^T + \boldsymbol{Q}_{t,t_P} \end{aligned}\right\} (P) \tag{2}$$

$$\left.\begin{aligned} &\left\{\boldsymbol{i}_{s,t} = \boldsymbol{H}_{s,t}^T\boldsymbol{R}_{s,t}^{-1}\boldsymbol{\xi}_{s,t,t}, \boldsymbol{I}_{s,t} = \boldsymbol{H}_{s,t}^T\boldsymbol{R}_{s,t}^{-1}\boldsymbol{H}_{s,t}\right\} (\perp_M) \\ &\left\{\boldsymbol{i}_t = \sum_{s=1}^{S}\boldsymbol{i}_{s,t}, \boldsymbol{I}_t = \sum_{s=1}^{S}\boldsymbol{I}_{s,t}\right\} (+) \\ &\left\{\hat{\boldsymbol{y}}_{t|t} = \hat{\boldsymbol{y}}_{t|t_P} + \boldsymbol{i}_t, Y_{t|t} = \boldsymbol{Y}_{t|t_P} + \boldsymbol{I}_t\right\} (A) \end{aligned}\right\}. \tag{3}$$

To deal with erroneous measurements, a validation test that checks if $\boldsymbol{\xi}_{s,k,a}$ is coherent with the estimate of the state is sometimes included before the KF/IF update step (Hall, 1992). If the test is passed, $\boldsymbol{\xi}_{s,k,a}$ is used to update the estimate, otherwise it is rejected. Data association distances (Fukunaga, 1990) are used as validation tests because they are quick geometric methods to quantify the disagreement that exists between $\boldsymbol{\xi}_{s,k,a}$ and its predicted value $(\boldsymbol{H}_{s,k}\hat{\boldsymbol{x}}_{k|k_P})$. A typical validation test consists in comparing the obtained distance with a threshold l_s. The Mahalanobis distance $d_{s,k,a}$ in Eq. (4) is often used because it weighs the discrepancy between $\boldsymbol{\xi}_{s,k,a}$ and $\boldsymbol{H}_{s,k}\hat{\boldsymbol{x}}_{k|k_P}$ with the inverse of the covariance of the predicted measurement value $(\boldsymbol{H}_{s,k}\boldsymbol{P}_{k|k_P}\boldsymbol{H}_{s,k}^T + \boldsymbol{R}_{s,k})$. Thus, it grows with the discrepancy and decreases with the uncertainty. Further, $d_{s,k,a}$ follows a chi-square distribution $\chi^2_{n_s}$ of as many degrees of freedom n_s as the number of elements in $\boldsymbol{\xi}_{s,k,a}$. Consequently, the validation test only rejects valid measurements with a probability lower than α when the cumulative probability $P(\chi^2_{n_s} < l_s) = 1 - \alpha/2$. See Johnson and Wichern (1998), for further details.

$$\begin{aligned} &d_{s,k,a} \leq l_s \\ &d_{s,k,a} = \boldsymbol{e}_{s,k,a}^T(\boldsymbol{H}_{s,k}\boldsymbol{P}_{k|k_P}\boldsymbol{H}_{s,k}^T + \boldsymbol{R}_{s,k})^{-1}\boldsymbol{e}_{s,k,a} \\ &\boldsymbol{e}_{s,k,a} = (\boldsymbol{\xi}_{s,k,a} - \boldsymbol{H}_{s,k}\hat{\boldsymbol{x}}_{k|k_P}). \end{aligned} \tag{4}$$

[1] Note that when t_P is substituted by $t-1$, the first expression of Eq. (1) becomes $\boldsymbol{x}_t = \boldsymbol{F}_{t,t-1}\boldsymbol{x}_{t-1} + \boldsymbol{u}_{t,t-1} + \boldsymbol{\upsilon}_{t,t-1}$, which is the usual equation in discrete linear systems, used to present IFAsyn-I in Besada-Portas et al. (2009, 2007). Our new notation better suits the aperiodicity of the events supported by the versions of IFAsyn introduced in this paper. The evaluation of $\boldsymbol{F}_{t,t_P}$, $\boldsymbol{u}_{t,t_P}$, $\boldsymbol{Q}_{t,t_P}$ depends on the system. For instance, when it is a discretized version of a continuous system modeled by $\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{u}(t) + \boldsymbol{\upsilon}(t)$, the three variables can be calculated as $\boldsymbol{u}_{t,t_P} = \int_0^{t-t_P}\boldsymbol{\phi}(\tau)\boldsymbol{u}(\tau)\mathrm{d}\tau$, $\boldsymbol{F}_{t,t_P} = \phi(t-t_P)$ and $\boldsymbol{Q}_{t,t_P} = \int_0^{t-t_P}\boldsymbol{\phi}(\tau)\boldsymbol{Q}(\tau)\boldsymbol{\phi}^T(\tau)\mathrm{d}\tau$, where $\boldsymbol{x}(t)$ and $\dot{\boldsymbol{x}}(t)$ are the continuous state and its time derivate at time t, $\boldsymbol{A}$ the transition matrix, $\boldsymbol{u}(t)$ the control signal at t, $\boldsymbol{Q}(t)$ the covariance of the continuous zero mean noise $\boldsymbol{\upsilon}(t)$, and $\boldsymbol{\phi}(t) = \mathrm{e}^{\boldsymbol{A}t}$.

$$\begin{array}{ccccc} \boldsymbol{\hat{y}}_{j|j} & = & \boldsymbol{\hat{y}}_{j|j_P} & + & \sum_{s\in \mathrm{Av}(j)} \boldsymbol{i}_{s,j} \\ \boldsymbol{Y}_{j|j} & = & \boldsymbol{Y}_{j|j_P} & + & \sum_{s\in \mathrm{Av}(j)} \boldsymbol{I}_{s,j} \\ \text{Information state with all data up to } j & & \text{Information state with all data before } j & & \text{Measurements at } j \text{ (projected into the information space)} \\ \{\boldsymbol{\xi}_{s,k\le j,a\le t}\} & & \{\boldsymbol{\xi}_{s,k<j,a\le t}\} & & \{\boldsymbol{\xi}_{s,j,a\le t}\} \end{array}$$

Fig. 1. Division in the IF update step between the sensorial information at j and the previous one. $\mathrm{Av}(j) = \{s|\boldsymbol{\xi}_{s,j,a\le t}\}$ stands for the set of sensors whose measurements at time j are already available at the current time t.

2.3. Estimating the state with OOS data ($\boldsymbol{\xi}_{s,k,t>k}$)

When some delayed data arrive at the fusion center ($\boldsymbol{\xi}_{s,k,t>k}$), the KF and IF cannot use it to update the current ($\boldsymbol{\hat{x}}_{t|t}, \boldsymbol{P}_{t|t}$). The new developed algorithms will be equivalent to the KF and IF when their results with OOS data are the same as the results of the KF and IF with the same data without delays.

It is important to highlight that their equivalency will only hold for open control systems, that is when $\boldsymbol{u}_t$ does not depend on $\boldsymbol{x}_t$. The dependency $\boldsymbol{u}_t = f(\boldsymbol{x}_t)$ that exists in closed loop controllers will break that equivalency when any $\boldsymbol{u}_t$ is calculated before a delayed $\boldsymbol{\xi}_{s,k,a}$ ($k \le t$, $a > t$) is used to update the values of ($\boldsymbol{\hat{x}}_{t|t}, \boldsymbol{P}_{t|t}$).

3. IFAsyn: an IF for the OOSP

3.1. Fundamentals

The origin of IFAsyn is the Junction Tree Algorithm (JTA) presented in Lauritzen (1992), which is an exact inference algorithm for Bayesian Networks (BNs) (Cowell, Dawid, Lauritzen, & Spiegelhalter, 1999) with continuous random variables and linear Gaussian dependencies that estimates the state of any set of variables of the BN given the data. As the order in which the data are introduced does not change the final estimates, the JTA is itself an optimal solution for estimating the state of the variables of any OOSP with continuous random variables and linear Gaussian dependencies. Therefore, the JTA can directly solve our problem, although it spends too many memory and computational resources. To minimize them, we analyzed how it works in detail in Besada-Portas (2004), taking into account its equivalence with the IF when the measurements are not delayed (Murphy, 1998), and conclude that the JTA works for the OOSP because it separates the information that has already been propagated from the new incoming one.

This type of division also appears naturally in the IF update step (3), as Fig. 1 highlights. The addition (+) separates the estimate with all the sensorial information previous to j ($\boldsymbol{\hat{y}}_{j|j_P}, \boldsymbol{Y}_{j|j_P}$) from the sensorial information of instant j which is already available, projected into the information space and accumulated ($\boldsymbol{i}_j = \sum_{s\in \mathrm{Av}(j)} \boldsymbol{i}_{s,j}, \boldsymbol{I}_j = \sum_{s\in \mathrm{Av}(j)} \boldsymbol{I}_{s,j}$). The separation can be exploited by (1) storing ($\boldsymbol{\hat{y}}_{j|j_P}, \boldsymbol{Y}_{j|j_P}, \boldsymbol{i}_j, \boldsymbol{I}_j$) for each time step j, and (2) updating the values of ($\boldsymbol{i}_j, \boldsymbol{I}_j$) and $\{\boldsymbol{\hat{y}}_{k|k_P}, \boldsymbol{Y}_{k|k_P} \,|\, \forall k > j\}$ when any $\boldsymbol{\xi}_{s,j,a}$ arrives at the fusion center. The new values of ($\boldsymbol{i}_j, \boldsymbol{I}_j$) are easily calculated with a measurement projection (3)($\perp_M$) and accumulation (3)(+). For each k, updating the values of $\{\boldsymbol{\hat{y}}_{k|k_P}, \boldsymbol{Y}_{k|k_P}\}$ requires a measurement assimilation (3)(A), a projection ($\perp_S$) into the state space, a prediction (2)(P), and a projection ($\perp_S$) into the information space. The predictions (2)(P) also need the control signal $\boldsymbol{u}_{j_N,j}$ from each j to the next j_N, and thus for each time step ($\boldsymbol{\hat{y}}_{j|j_P}, \boldsymbol{Y}_{j|j_P}, \boldsymbol{i}_j, \boldsymbol{I}_j, \boldsymbol{u}_{j_N,j}$) need to be stored.

3.2. IFAsyn-(I, II, III): IFAsyn without validation step

IFAsyn-I, the first version of IFAsyn presented in Besada-Portas (2004) and Besada-Portas et al. (2009, 2007), was developed for

```
SData($\boldsymbol{\xi}_{s,k,a}, LI, k_U$)
//$\boldsymbol{\xi}_{s,k,a}$ (in): Measurement of sensor $s$ at time $k$
//$LI$ (in/out): Time ordered list with information data
//$k_U$ (in/out): Time up to with $LI$ has its info updated

$\boldsymbol{i}_{s,k} = \boldsymbol{H}^T_{s,k}\boldsymbol{R}^{-1}_{s,k}\boldsymbol{\xi}_{s,k,a}$;  $\boldsymbol{I}_{s,k} = \boldsymbol{H}^T_{s,k}\boldsymbol{R}^{-1}_{s,k}\boldsymbol{H}_{s,k}$; //3.($\perp_M$)
//Get data from the list for $k$ or previous $k_P$
$k_P = LI$.Get($k, \boldsymbol{\hat{y}}, \boldsymbol{Y}, \boldsymbol{i}, \boldsymbol{I}$);
if ($k == k_P$) then //There is information for $k$
  | $\boldsymbol{i} = \boldsymbol{i} + \boldsymbol{i}_{s,k}$;  $\boldsymbol{I} = \boldsymbol{I} + \boldsymbol{I}_{s,k}$;  //3.(+)
else //There is no data for $k$: initialize it
  | $\boldsymbol{i} = \boldsymbol{i}_{s,k}$; $\boldsymbol{I} = \boldsymbol{I}_{s,k}$; $\boldsymbol{\hat{y}} = \emptyset$; $\boldsymbol{Y} = \emptyset$ //No data for $\boldsymbol{\hat{y}}, \boldsymbol{Y}$
end
$LI$.Set($k, \boldsymbol{\hat{y}}, \boldsymbol{Y}, \boldsymbol{i}, \boldsymbol{I}$); //Set data for $k$
$k_U = \min(k_P, k_U)$;  //$LI$ updated up to $k_U$
```

Fig. 2. SData: projecting $\boldsymbol{\xi}_{s,j,a}$, accumulating its information, and storing it in LI.

```
CData($\boldsymbol{u}_k, LC, LI, k_U$)
//$\boldsymbol{u}_k$ (in): Control signal applied from $k$
//$LC$ (in/out): Time ordered list with control functions
//$LI$ (in/out): Time ordered list with information data
//$k_U$ (in/out): Time up to with $LI$ has its info updated

$LC$.Set($k, \boldsymbol{u}_k$); //Store control function $\boldsymbol{u}_k$ in $LC$
$k_P = LI$.Get($k, \boldsymbol{\hat{y}}, \boldsymbol{Y}, \boldsymbol{i}, \boldsymbol{I}$);
if ($k! = k_P$) then //No information for $k$: initilize it
  | //no data for ($\boldsymbol{\hat{y}}_k, \boldsymbol{Y}_k$), $\boldsymbol{i}_k = \boldsymbol{0}, \boldsymbol{I}_k = \boldsymbol{0}$
  | $LI$.Set($k, \emptyset, \emptyset, \boldsymbol{0}, \boldsymbol{0}$);
end
$k_U = \min(k_P, k_U)$;  //Last updated time $k_U$
```

Fig. 3. CData: incorporating the entries in LC & LI for the new control function.

systems where the time-stamp of all the possible $\boldsymbol{\xi}_{s,j,a}$ is known beforehand. Its prediction step combined the KF prediction (2) with the initialization of the projection measurement variables ($\boldsymbol{i}_k = \boldsymbol{0}, \boldsymbol{I}_k = \boldsymbol{0}$). Its update step implemented the idea presented in Section 3.1 exactly. It stored the five variables for all the time-steps where a measurement can be taken and a control signal changed. Finally, the re-propagation cycle could go through time-steps without previously available $\boldsymbol{\xi}_{s,j,a}$, overloading the system unnecessarily.

The novel versions of IFAsyn presented in this section do not require the previous knowledge of the time-stamp of $\boldsymbol{\xi}_{s,j,a}$ and accommodate its memory and computation requirements to the available information. They reorganize IFAsyn-I steps and combine them with the management of two dynamic lists, LC and LI, that respectively store the control functions $\boldsymbol{u}_j$ starting at j, and the information variables ($\boldsymbol{\hat{y}}_{j|j_P}, \boldsymbol{Y}_{j|j_P}, \boldsymbol{i}_j, \boldsymbol{I}_j$) for the time-stamps associated with changes of the control functions $\boldsymbol{u}_j$ and already assimilated $\boldsymbol{\xi}_{s,j,a}$. They are controlled by the next functions. $k_N = LI.\mathrm{Next}(k)$ returns the time-stamp of the entry of LI whose time comes after k (-1 if it does not exist). $k_R = LI.\mathrm{Get}(k, \boldsymbol{y}, \boldsymbol{Y}, \boldsymbol{i}, \boldsymbol{I})$ returns ($\boldsymbol{y}, \boldsymbol{Y}, \boldsymbol{i}, \boldsymbol{I}$) for time k if they exist or for the closest previous time if they do not, plus the time k_R of the returned information. $LI.\mathrm{Set}(k, \boldsymbol{y}, \boldsymbol{Y}, \boldsymbol{i}, \boldsymbol{I})$ sets in LI the values of time-stamp k to ($\boldsymbol{y}, \boldsymbol{Y}, \boldsymbol{i}, \boldsymbol{I}$), modifying the stored values if there is already an entry for k or creating a new one if it does not. $LC.\mathrm{Set}(k, u_k)$ stores the control function starting at k. $\boldsymbol{u}_{k_2,k_1} = LC.\mathrm{Get}(k_2, k_1)$ calculates and returns the control signal $\boldsymbol{u}_{k_2,k_1}$ applied from k_1 to k_2.[2]

The new versions of IFAsyn are built over 3 functions, SData, CData and Update&Get, whose steps are detailed in pseudo-code in Figs. 2–4. SData projects the new $\boldsymbol{\xi}_{s,j,a}$ (3)($\perp_M$), accumulates it to others related to the same time-stamp (3)(+), and stores it in LI. CData incorporates the new $\boldsymbol{u}_k$ that starts at k in LC and creates

[2] LC can also store the requested $\boldsymbol{u}_{k_2,k_1}$ to have them available when they are needed again. Its storage requirements can be minimized deleting all $\boldsymbol{u}_{k_j,k_i}$ whose time intervals partially coincide with the interval of the newly required $\boldsymbol{u}_{k_2,k_1}$.

```
Update&Get($k_C, LC, LI, k_U, \hat{x}_{k_C|k_C}, P_{k_C|k_C}$)
//$k_C$ (in): Update variables up to time $k_C$
//$LC$ (in/out): Time ordered list with control functions
//$LI$ (in/out): Time ordered list with information data
//$k_U$ (in/out): Time up to with $LI$ has its info updated
//$\hat{x}_{k_C|k_C}, P_{k_C|k_C}$ (out): updated state and cov. at $k_C$

$k = \min(k_U, k_C)$; //Repropagate or get state at $k_C$
//Get data for $k_U$, or $k_C$ or the previous time to $k_C$
$k = LI.\mathrm{Get}(k, \hat{y}_{k|k_P}, Y_{k|k_P}, i_k, I_k)$;
while ($k <= k_C$)
  $\hat{y}_{k|k} = \hat{y}_{k|k_P} + i_k$;  $Y_{k|k} = Y_{k|k_P} + I_k$;  //3.(A)
  $\hat{x}_{k|k} = Y^{-1}_{k|k}\hat{y}_{k|k}$;  $P_{k|k} = Y^{-1}_{k|k}$;  //($\perp_S$)
  if ($k < k_C$) then //Keep on predicting up to $k_C$
    $k_F = LI.\mathrm{Next}(k)$; //Time of following entry in $LI$
    //$k_N$ (time after prediction): when there is no entry in
    //$LI$ for $k_C$, at some point $k_F = -1$ (end of $LI$)
    //or $k_F > k_C$. In these cases, $k_N = k_C$
    if ($k_F == -1$) $\vee$ ($k_F > k_C$) then $k_N = k_C$;
    else $k_N = k_F$; end
    $\hat{x}_{k_N|k} = F_{k_N,k}\hat{x}_{k|k} + LC.Get(k_N, k)$   } //2.(P)
    $P_{k_N|k} = F_{k_N,k}P_{k|k}F^T_{k_N,k} + Q_{k_N,k}$         }
    $\hat{y}_{k_N|k} = P^{-1}_{k_N|k}\hat{x}_{k_N|j}$;  $Y_{k_N|k} = P^{-1}_{k_N|k}$;  //($\perp_S$)
    //Update local variables for nex iteration
    $k_P = k$;  $k = k_N$;  $\hat{y}_{k|k_P} = \hat{y}_{k_N|k}$;  $Y_{k|k_P} = Y_{k_N|k}$;
    if ($k_F == -1$)$\vee$($k_F > k_C$) then //No info in $LI$ at $k_C$
      $i_k = 0$; $I_k = 0$;  //No sensorial info in $LI$ at $k_C$
    else //Data in $LI$ for $k$, Get and Set information
      $LI.\mathrm{Get}(k, y, Y, i_k, I_k)$; //Get $i_k, I_k$, throw away $y$,$Y$
      $LI.\mathrm{Set}(k, \hat{y}_{k|k_P}, Y_{k|k_P}, i_k, I_k)$; //Set $\hat{y}_{k|k_P}$,$Y_{k|k_P}$
      $k_U = k$;  //$LI$ updated up to $k$
    end
  end
end
$\hat{x}_{k_C|k_C} = \hat{x}_{k|k}$; $P_{k_C|k_C} = P_{k|k}$;  //Return estimate at $k_C$
```

Fig. 4. Update&Get: updating *LI* entries with the stored data and returning the estimates, up to the given time k_C.

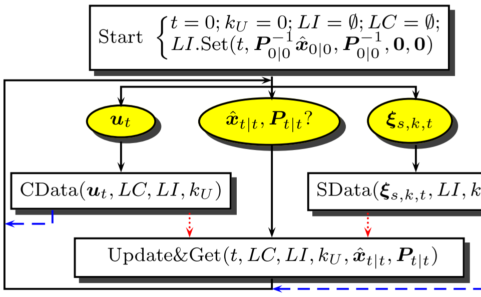

Start $\begin{cases} t = 0; k_U = 0; LI = \emptyset; LC = \emptyset; \\ LI.\mathrm{Set}(t, P^{-1}_{0|0}\hat{x}_{0|0}, P^{-1}_{0|0}, 0, 0) \end{cases}$
u_t
$\hat{x}_{t|t}, P_{t|t}$?
$\xi_{s,k,t}$
CData(u_t, LC, LI, k_U)
SData($\xi_{s,k,t}, LI, k_U$)
Update&Get($t, LC, LI, k_U, \hat{x}_{t|t}, P_{t|t}$)

Fig. 5. Flowcharts of IFAsyn-II & IFAsyn-III. Solid black arrows are common to both algorithms, red dotted ones refer to IFAsyn-II and blue dashed ones refer to IFAsyn-III. Ovals represent events and boxes show the functions to be run. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

a new entry in *LI* for that instant if it does not exist. Update&Get re-propagates the already projected and accumulated sensorial information through *LI* up to a given time k_C using a series of assimilations (3)(A), predictions (2)(P) and projections ($\perp_S$), and returns ($\hat{\mathbf{x}}_{k_C|k_C}, \mathbf{P}_{k_C|k_C}$). SData and CData modify the number of list entries when needed, while Update&Get only changes the contents of their elements.[3] k_U identifies up to which time-stamp the *LI* entries have their ($\hat{\mathbf{y}}_{k|k_P}, \mathbf{Y}_{k|k_P}$) already updated with the available data.

The new IFAsyn-II and IFAsyn-III are depicted in the two flowcharts represented simultaneously in Fig. 5, where the yellow ovals show a change in $\mathbf{u}_t$, a requirement of the current ($\hat{\mathbf{x}}_{t|t}, \mathbf{P}_{t|t}$),

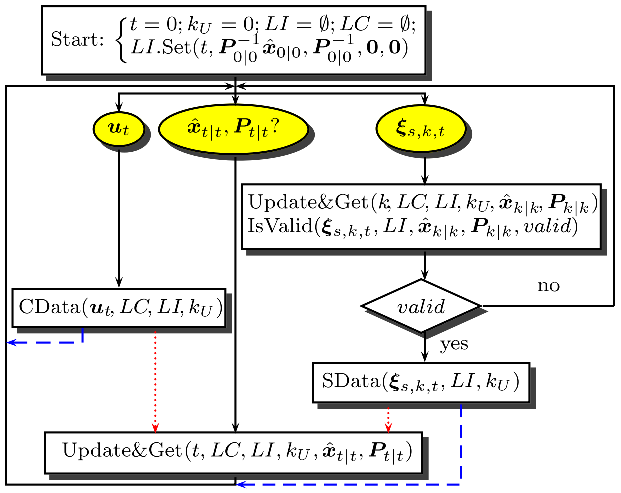

Start: $\begin{cases} t = 0; k_U = 0; LI = \emptyset; LC = \emptyset; \\ LI.\mathrm{Set}(t, P^{-1}_{0|0}\hat{x}_{0|0}, P^{-1}_{0|0}, 0, 0) \end{cases}$
u_t
$\hat{x}_{t|t}, P_{t|t}$?
$\xi_{s,k,t}$
Update&Get($k, LC, LI, k_U, \hat{x}_{k|k}, P_{k|k}$)
IsValid($\xi_{s,k,t}, LI, \hat{x}_{k|k}, P_{k|k}, valid$)
no
valid
yes
CData(u_t, LC, LI, k_U)
SData($\xi_{s,k,t}, LI, k_U$)
Update&Get($t, LC, LI, k_U, \hat{x}_{t|t}, P_{t|t}$)

Fig. 6. Flowcharts of IFAsyn-IV & IFAsyn-V. Solid black arrows are common to both algorithms, red dotted ones refer to IFAsyn-IV and blue dashed ones refer to IFAsyn-V. Ovals represent events and boxes show the functions to be run. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

and a $\boldsymbol{\xi}_{s,k,t}$ arrival; the boxes indicate the functions to be carried out, and the colors and arrow types identify to which filter they are related. IFAsyn-II, whose flowchart contains the black and red dotted arrows, calculates ($\hat{\mathbf{x}}_{t|t}, \mathbf{P}_{t|t}$) with all the available information whenever there is a new $\boldsymbol{\xi}_{s,k,t}$ or $\mathbf{u}_t$, or when it is required by an external event. Hence, it calls Update&Get after including any new $\boldsymbol{\xi}_{s,k,t}$ or $\mathbf{u}_t$ with SData or CData. IFAsyn-III, whose flowchart contains the black and blue dashed arrows, minimizes the computational cost postponing the re-propagation of the new $\boldsymbol{\xi}_{s,k,t}$ or $\mathbf{u}_t$ until ($\hat{\mathbf{x}}_{t|t}, \mathbf{P}_{t|t}$) are required by an external event. The speed-up is achieved when at least two pieces of data ($\boldsymbol{\xi}_{s,k,t}$ and/or $\mathbf{u}_t$) are stored by SData and/or CData before calling Update&Get. Finally, when we incorporate a new $\mathbf{u}_k$ for all the time-stamps of the possible $\boldsymbol{\xi}_{s,k,t}$, IFAsyn-II behaves as IFAsyn-I.

IFAsyn-(I, II, III) are optimal MMSE algorithms for our OOSP because the order of their operations makes them roll back to the time-stamp of the older piece of data (k_U) and re-start the fusion process with the sensorial information already projected into the information space. Therefore, they obtain the same results with OOS data as the KF with the same data without delays.

3.3. IFAsyn-(IV, V): IFAsyn with validation step

Including a validation step based on a distance between $\boldsymbol{\xi}_{s,k,t}$ and its estimated value in IFAsyn lets it detect and reject corrupted measurements whose errors are not modeled by $\mathbf{R}_{s,k}$. Although its inclusion before the update operation of the KF is straightforward, executing it in IFAsyn requires considering some important aspects that affect the final results of the filters. Our filters with validation step are named IFAsyn-IV and IFAsyn-V and are depicted in the flowcharts in Fig. 6, IFAsyn-IV with black and red dotted arrows, and IFAsyn-V with black and blue dashed ones.

First, the assimilation of any OOS $\boldsymbol{\xi}_{s,k,t}$ changes ($\hat{\mathbf{x}}_{j|j}, \mathbf{P}_{j|j}$) for all $j \geq k$. This could modify the results of the validation tests that the $\boldsymbol{\xi}_{s,l,a}$ with a bigger time-stamp ($l > k$) which arrived at the fusion center earlier ($a < t$) has already passed. Although redoing the validation tests to incorporate the new valid measurements and to eliminate the new invalid ones will let our filters obtain the same results as KF or IF, the prohibitive computational cost associated with it makes us reject this option. Consequently, our filters carry out the validation test of each $\boldsymbol{\xi}_{s,k,t}$ only once before incorporating it into the projected sensorial information with SData. Additionally,

[3] When *LC* also stores $\mathbf{u}_{k_i,k_j}$, Update&Get can modify the number of $\mathbf{u}_{k_i,k_j}$ entries in *LC* when calling *LC.Get*(k_2, k_1).

```
IsValid($\boldsymbol{\xi}_{s,k,a}, LI, \hat{\boldsymbol{x}}_{k|k}, \boldsymbol{P}_{k|k}, valid$)
//$\xi_{s,k,a}$ (in): Measurement of sensor $s$ at time $k$
//$LI$ (in): Time ordered list with information data
//$\hat{\boldsymbol{x}}_{k|k}, \boldsymbol{P}_{k|k}$ (input): updated state and covariance at $k$
//$valid$ (out): true if $\xi_{s,k,a}$ is valid

//We assume that Update&Get has been called before
$k_P = LI.\mathrm{Get}(k, \hat{\boldsymbol{y}}_{k|k_P}, \boldsymbol{Y}_{k|k_P}, \boldsymbol{i}_k, \boldsymbol{I}_k)$;
$if\ (k == k_P)\ then$ //There is an entry in $LI$ for $k$
    $\hat{\boldsymbol{x}}_{k|k_P} = \boldsymbol{Y}^{-1}_{k|k_P}\hat{\boldsymbol{y}}_{k|k_P}$; $\boldsymbol{P}_{k|k_P} = \boldsymbol{Y}^{-1}_{k|k_P}$; //$(\perp_S)$
$else$ //There is no info in $LI$ for $k$
    $\hat{\boldsymbol{x}}_{k|k_P} = \hat{\boldsymbol{x}}_{k|k}$; $\boldsymbol{P}_{k|k_P} = \boldsymbol{P}_{k|k}$;
$end$
//Do a validation test, for example (4)
$\boldsymbol{e}_{s,k,a} = \left(\boldsymbol{\xi}_{s,k,a} - \boldsymbol{H}_{s,k}\hat{\boldsymbol{x}}_{k|k_P}\right)$
$valid = \boldsymbol{e}^T_{s,k,a}\left(\boldsymbol{H}_{s,k}\boldsymbol{P}_{k|k_P}\boldsymbol{H}^T_{s,k} + \boldsymbol{R}_{s,k}\right)^{-1}\boldsymbol{e}_{s,k,a} \le l_s$
```

Fig. 7. IsValid: validating if a measurement is valid.

in order to minimize the influence of the OOS reception of the data, the test for $\boldsymbol{\xi}_{s,k,t}$ is carried out with the already updated state and covariance at k, and thus IFAsyn-IV and IFAsyn-V call Update&Get before the validation step function IsValid.

Second, when the OOS $\boldsymbol{\xi}_{s,k,t}$ has the same time-stamp as an already validated and assimilated $\boldsymbol{\xi}_{s,k,a}$ $(t > a)$ the validation of $\boldsymbol{\xi}_{s,k,t}$ can be done using the estimate of the state and covariance that already includes $\boldsymbol{\xi}_{s,k,a}$ $(\boldsymbol{x}_{k|k}, \boldsymbol{P}_{k|k})$ or the state and covariance that does not $(\boldsymbol{x}_{k|k_P}, \boldsymbol{P}_{k|k_P})$. We opt for the second choice[4] because the original validation test (Eq. (4)) is carried out according to it and because by not considering $\boldsymbol{\xi}_{s,k,a}$ to validate $\boldsymbol{\xi}_{s,k,t}$, the validation step evicts to give a higher validation priority for a given time-stamp to the measurements that have arrived first.

IFAsyn-IV and IFAsyn-V extend IFAsyn-II and IFAsyn-III with a validation step, using the pseudo-code of Fig. 7 for IsValid. Including Update&Get before IsValid to reduce the effects of the OOS arrival of $\boldsymbol{\xi}_{s,k,t}$ in IsValid, makes IFAsyn-V lose part of IFAsyn-III speed-up. As none of these re-validates the already assimilated $\boldsymbol{\xi}_{s,l,a}$, their results with OOS data can differ from the ones from KF with the same data without delays. However, when the filters are well tuned the re-validations should almost not vary and the results of our filters and KF should be similar. When all the filters, including KF, are badly tuned, they can accept non-valid $\boldsymbol{\xi}_{s,k,t}$ and return incorrect results. The OOSP either alleviates or aggravates the problem depending on the arrival order.

Finally, to further reduce the memory requirements of any IFAsyn, we can use another operation that deletes those entries that are outside a time window from LI and LU, at the expense of not being able to assimilate those $\boldsymbol{\xi}_{s,k,t}$ whose time-stamps are not longer included.

4. Related work

In this section we describe many other OOS algorithms and compare them with IFAsyn. Considering their core idea and some of the assumptions made during its development, we can classify them into the following 4 groups:

Group 1, (Anxi et al., 2005; Bar-Shalom, 2002; Bar-Shalom et al., 2004; Hilton et al., 1993; Ito et al., 1998; Mallick et al., 2001): the retrodiction KF algorithms use the transition model (1) to backwardly retrieve the state at the $\boldsymbol{\xi}_{s,k,t}$ time-stamp. They invert $\boldsymbol{F}_{t,k}$, that makes them valid only for discrete systems with non-singular $\boldsymbol{F}_{t,k}$ (a property that holds for all discretized continuous systems). They have low memory and computational needs, and all except Bar-Shalom (2002) are always suboptimal. The optimality of Bar-Shalom (2002) only holds when $\boldsymbol{\xi}_{s,k,t}$ is not older than any already assimilated $\boldsymbol{\xi}_{s,l,a}$ $(k \ge l)$ (i.e, when $\boldsymbol{\xi}_{s,k,t}$ are delayed only one time lag), and Bar-Shalom et al. (2004) is usually considered the standard suboptimal solution.

Group 2, (Challa et al., 2002; Feng et al., 2008; Lu et al., 2005; Matveev & Savkin, 2003; Zhang et al., 2004): KF algorithms that modify the model (1) and use non-delayed KF equations to assimilate any $\boldsymbol{\xi}_{s,k,t}$. First, Challa et al. (2002) and Matveev and Savkin (2003) augment the state to include the last W states and modify the transition and measurements models (Eq. (1)) to consider the state extension. In order to calculate a value of W that lets them assimilate all $\boldsymbol{\xi}_{s,k,t}$, they need to know (1) the value of the maximum permitted delay and (2) the time-stamp of every $\boldsymbol{\xi}_{s,k,t}$, to ensure that there is a slot available for each measurement time step k in the extended state system. Challa et al. (2002) uses the normal KF in the extended model and thus it operates in a large state space, while Matveev and Savkin (2003) exploits the sparse structure of the extended transition and measurement matrices to obtain a new set of expressions and thus loses the simplicity of Challa et al. (2002). Second, Feng et al. (2008) augments the measurement space, modifies the measurement equations in (1), and assimilates $\boldsymbol{\xi}_{s,k,t}$ with the KF for systems with correlated measurement and state noise. Its optimality only holds when the $\boldsymbol{\xi}_{s,k,t}$ delays are no bigger than one time step. Third, Lu et al. (2005) and Zhang et al. (2004) readjust their models with a reorganization of the innovation sequence that makes them directly apply the third naïve solution.

Group 3, (Zhang et al., 2002, 2005): KF algorithms that pre-calculate sets of variables to be ready to update the current state as soon as any $\boldsymbol{\xi}_{s,k,t}$ arrives. They need to know beforehand the time-stamp of all the possible $\boldsymbol{\xi}_{s,k,t}$ to have sets of variables ready for each of them. Further, the pre-calculation of the sets of variables has to be re-started after each $\boldsymbol{\xi}_{s,k,t}$ is assimilated. Finally, their inherent complexity makes any modification difficult.

Group 4, (Lanzkron & Bar-Shalom, 2004; Nettleton & Durrant-Whyte, 2001; Rheaume & Benaskeur, 2008; WenHui et al., 2006; Zhang et al., 2003): For each time step they store $(\hat{\boldsymbol{x}}_{j|j}, \boldsymbol{P}_{j|j})$ or $(\hat{\boldsymbol{y}}_{j|j}, \boldsymbol{Y}_{j|j})$, and forward the information of any new $\boldsymbol{\xi}_{s,k,t}$ to sequentially update all the states and covariances from the time-stamp of $\boldsymbol{\xi}_{s,k,t}$ (Nettleton & Durrant-Whyte, 2001; WenHui et al., 2006; Zhang et al., 2003) or only the current ones (Lanzkron & Bar-Shalom, 2004; Rheaume & Benaskeur, 2008). Besides, Nettleton and Durrant-Whyte (2001) is really close to IFAsyn, although it avoids state projections $(\perp_S)$ with an IF prediction that inverts $\boldsymbol{F}_{j,j_P}$. Accordingly, Nettleton and Durrant-Whyte (2001) is not valid for some discrete systems and, as we want to estimate $(\hat{\boldsymbol{x}}_{k|k}, \boldsymbol{P}_{k|k})$ we have to include $(\perp_S)$ in it as well. Zhang et al. (2003) is more complex than IFAsyn because instead of updating $(\hat{\boldsymbol{x}}_{j|j}, \boldsymbol{P}_{j|j})$ directly it updates some extra variables first and $(\hat{\boldsymbol{x}}_{j|j}, \boldsymbol{P}_{j|j})$ later. A close inspection of WenHui et al. (2006) shows that it does the same operations as IFAsyn substituting $(\boldsymbol{i}_j, \boldsymbol{I}_j)$ for $(\hat{\boldsymbol{y}}_{j|j} - \hat{\boldsymbol{y}}_{j|j_P}, \boldsymbol{Y}_{j|j} - \boldsymbol{Y}_{j|j_P})$ and the values in the information space for the projected ones in the state space. Finally, Lanzkron and Bar-Shalom (2004) and Rheaume and Benaskeur (2008) are the closest approximated algorithms to ours. When a new $\boldsymbol{\xi}_{s,k,t}$ arrives, Lanzkron and Bar-Shalom (2004) and Rheaume and Benaskeur (2008) substitute the update step in IFAsyn for: (1) two direct predictions (one with the new $\boldsymbol{\xi}_{s,k,t}$ and another without it) of $(\hat{\boldsymbol{y}}_{k|k}, \boldsymbol{Y}_{k|k})$ to the actual time t, and (2) an update of the values of $(\hat{\boldsymbol{y}}_{t|t}, \boldsymbol{Y}_{t|t})$ using their old values and the values obtained from the two predictions.

Finally, Shen et al. (2009) combines the exact retrodiction solution in Bar-Shalom (2002) and the forward update in Zhang et al. (2003) to develop a complex new optimal OOS algorithm, belonging to *groups* 1 and 4, that collects several delayed $\{\boldsymbol{\xi}_{s,k,t}\}$

[4] IsValid uses $(\boldsymbol{x}_{k|k}, \boldsymbol{P}_{k|k})$ when there is no entry in LI for k because in that case they are equal to $(\boldsymbol{x}_{k|k_P}, \boldsymbol{P}_{k|k_P})$.

Table 1
IFAsyn versus other algorithms. Column *ALG* identifies the algorithm; *SUP*, its support; F^{-1}, when it inverts F; $\boldsymbol{u}_t$, if it includes $\boldsymbol{u}_t$; *MS*, if it is multisensorial; *WK*, if it works without previous knowledge of the possible $\boldsymbol{\xi}_{s,k,t}$ time-stamps; *SI*, if it can assimilate multiple measurements in a single iteration, and *VAL*, if it includes a validation step. An '$\times$' means that it has that column property and a '$+$' that it can be modified to include it (Shen et al. (2009) can assimilate a group of $\boldsymbol{\xi}_{s,k,t}$ in a single iteration but needs to be modified to include a second batch). The 11 top filters are optimal and remain optimal when correctly modified to incorporate more properties except our validation step, while the 4 bottom ones are suboptimal.

ALG	*SUP*	F^{-1}	$\boldsymbol{u}_t$	*MS*	*WK*	*SI*	*VAL*
IFAsyn-I	IF		$\times$	$\times$	$+$	$+$	$+$
IFAsyn-II	IF		$\times$	$\times$	$\times$	$+$	$+$
IFAsyn-III	IF		$\times$	$\times$	$\times$	$\times$	$+$
Challa et al. (2002)	KF		$+$	$+$		$\times$	
Matveev and Savkin (2003)	KF		$\times$	$\times$		$\times$	
Zhang et al. (2002)-AI	KF		$+$	$+$			
Zhang et al. (2005)-AI	KF		$+$	$+$			
Nettleton and Durrant-Whyte (2001)	IF	$\times$	$\times$	$\times$	$+$	$+$	$+$
Zhang et al. (2003)-AI	KF		$+$	$+$	$+$	$+$	$+$
WenHui et al. (2006)	KF		$+$	$+$	$+$	$+$	$+$
Shen et al. (2009)	KF	$\times$	$+$	$\times$	$\times$	$\times/+$	
IFAsyn-IV	IF		$\times$	$\times$	$\times$	$+$	$\times$
IFAsyn-V	IF		$\times$	$\times$	$\times$	$\times$	$\times$
Bar-Shalom et al. (2004)	KF	$\times$	$+$	$\times$	$\times$		
Lanzkron and Bar-Shalom (2004) and Rheaume and Benaskeur (2008)	KF		$+$	$+$	$\times$	$+$	$+$

and assimilates them in a single iteration that only updates the final $(\hat{\boldsymbol{x}}_{t|t}, \boldsymbol{P}_{t|t})$. However, it cannot assimilate a second group of delayed $\{\boldsymbol{\xi}_{s,l,t'}\}$ with $l < t$ after the previous collection $\{\boldsymbol{\xi}_{s,k,t}\}$ has already been assimilated, unless it is modified to also update the intermediate $(\hat{\boldsymbol{x}}_{j|j}, \boldsymbol{P}_{j|j}, k < j < t)$.

Besides the properties that let us divide the algorithms into these 4 groups, the algorithms can also be distinguished by other characteristics. The most relevant ones are presented in Table 1, which includes all the optimal filters that can work with $\boldsymbol{\xi}_{s,k,t}$ delayed more than one time lag except Lu et al. (2005) and Zhang et al. (2004) (third naïve solution). It also includes two suboptimal filters whose degree of suboptimality is a small percentage of the Root Mean Square Error (RMSE): Bar-Shalom et al. (2004) as the standard one and Lanzkron and Bar-Shalom (2004) and Rheaume and Benaskeur (2008) as the closest to IFAsyn. The table shows that IFAsyn, which can be classified as *group* 4, is the current most general approach: IFAsyn-(I–V) do not invert $\boldsymbol{F}_{t,t_p}$, include $\boldsymbol{u}_{t,t_p}$ and work with multiple sensors; IFAsyn-(II–V) do not need to know beforehand the time-stamp of $\boldsymbol{\xi}_{s,k,t}$; IFAsyn-(III, V) assimilate groups of $\boldsymbol{\xi}_{s,k,t}$ in a single iteration; and IFAsyn-(IV, V) include a validation step.

IFAsyn is really easy to implement because it basically combines KF predictions, IF updates, and state projections. Its capacity of working without any prior knowledge of the $\boldsymbol{\xi}_{s,k,t}$ time-stamps is obtained only with the inclusion of two managed lists. Furthermore, it can assimilate multiple measurements $\boldsymbol{\xi}_{s,k,t}$ in a unique iteration with only postponing the update procedure. Its simplicity, which comes directly from its IF support, is only beaten by Challa et al. (2002) that directly uses the usual KF steps but needs to know beforehand the time-stamp of all $\boldsymbol{\xi}_{s,k,t}$.

Finally, an analysis of the stability of a control system that uses IFAsyn (or Challa et al. (2002), Nettleton and Durrant-Whyte (2001), Shen et al. (2009), WenHui et al. (2006), Zhang et al. (2003) and Zhang et al. (2002, 2005)) is out of the scope of this paper. On the one hand, IFAsyn-(I, II, III) are equivalent to Matveev and Savkin (2003) and therefore share their stability properties. In addition, although they assimilate the OOS data, the magnitude of the delays is important because while $\boldsymbol{\xi}_{s,k,t}$ is not available the control system works with intermittent observations that can instantaneously affect its stability (Hespanha et al., 2007; Sinopoli et al., 2004). The problem is not presented by IFAsyn-(I, II, III) and Challa et al. (2002), Matveev and Savkin (2003), Nettleton and Durrant-Whyte (2001), Shen et al. (2009), WenHui et al. (2006), Zhang et al. (2002, 2005) and Zhang et al. (2003), as they update the state as quickly as its computational requirements allow, but it has to be considered when they are used inside real-time control loops. On the other hand, IFAsyn-(IV, V) need to have a well-tuned validation step to properly assess the validity of $\boldsymbol{\xi}_{s,k,t}$. Although, a correct validation is also needed for the proper performance of the system that uses the KF with non-delayed data, OOS $\boldsymbol{\xi}_{s,k,t}$ affect the problem and special care should be taken.

4.1. Memory and computational cost comparison

In this section we analyze the memory and computational cost of the filters in Table 1, based on the following two assumptions: (1) since every filter can use $\boldsymbol{u}_{t,t_p}$ and multiple sensors, we modify the filters which do not use them originally to include them before performing the comparison, and (2) as some filters cannot easily incorporate the validation step, our comparison will not consider it, excluding IFAsyn-(IV, V). Besides, some filters need to know a priori the time-stamps of $\boldsymbol{\xi}_{s,k,t}$ and the window time W. Hence, we *initially* consider that they all work with the same prefixed time step, store the variables of the last W time-steps, and represent the control functions $\boldsymbol{u}_k$ with $\boldsymbol{u}_{k_N,k}$.

Table 2 shows that IFAsyn-(I, II, III) need more memory than Bar-Shalom et al. (2004), Lanzkron and Bar-Shalom (2004), Nettleton and Durrant-Whyte (2001), Rheaume and Benaskeur (2008), Shen et al. (2009), WenHui et al. (2006) and Zhang et al. (2003) because IFAsyn-(I, II, III) separate $(\hat{\boldsymbol{y}}_{j|j_P}, \boldsymbol{Y}_{j|j_P})$ from $(\boldsymbol{i}_j, \boldsymbol{I}_j)$ while Nettleton and Durrant-Whyte (2001) stores $(\hat{\boldsymbol{y}}_{j|j}, \boldsymbol{Y}_{j|j})$ and Bar-Shalom et al. (2004), Lanzkron and Bar-Shalom (2004), Rheaume and Benaskeur (2008), Shen et al. (2009), WenHui et al. (2006) and Zhang et al. (2003) store $(\hat{\boldsymbol{x}}_{j|j}, \boldsymbol{P}_{j|j})$. However, the extra memory in IFAsyn-(I, II, III) make them numerically more accurate. Dropping the last assumptions of the previous paragraph reduces the memory needs of IFAsyn-(II, III) and Shen et al. (2009): ours store $\boldsymbol{u}_j$ functions starting at j and $(\hat{\boldsymbol{y}}_{j|j_P}, \boldsymbol{Y}_{j|j_P}, \boldsymbol{i}_j, \boldsymbol{I}_j)$ for j related with already assimilated $\boldsymbol{\xi}_{s,j,t}$ and $\boldsymbol{u}_j$ changes, and Shen et al. (2009) stores $(\hat{\boldsymbol{x}}_{j|j_P}, \boldsymbol{P}_{j|j_P})$ and $\boldsymbol{u}_j$ for j related with $\boldsymbol{u}_j$ changes.

Table 3 shows the computational cost of the filters, calculated analytically counting[5] the number of FLoating Point Operations (FLOPs) used to carry out a prediction and assimilation of a set of m $\boldsymbol{\xi}_{s_i,k_i,t}$, each delayed r_i time-steps ($r_i = t - k_i$).[6] IFAsyn-(I, II)

[5] The count is carried out using LightSpeed Matlab Toolbox (http://research.microsoft.com/minka/software/lightspeed/) and symbolic variables with the matrices sizes. We minimize the number of operations of all the filters, counting an operation that is repeated in consecutive equations only once. We do not compute the number of FLOPs needed to obtain $\boldsymbol{u}_{t_N,t}$ because it depends on the problem and penalizes all the filters in the same way.

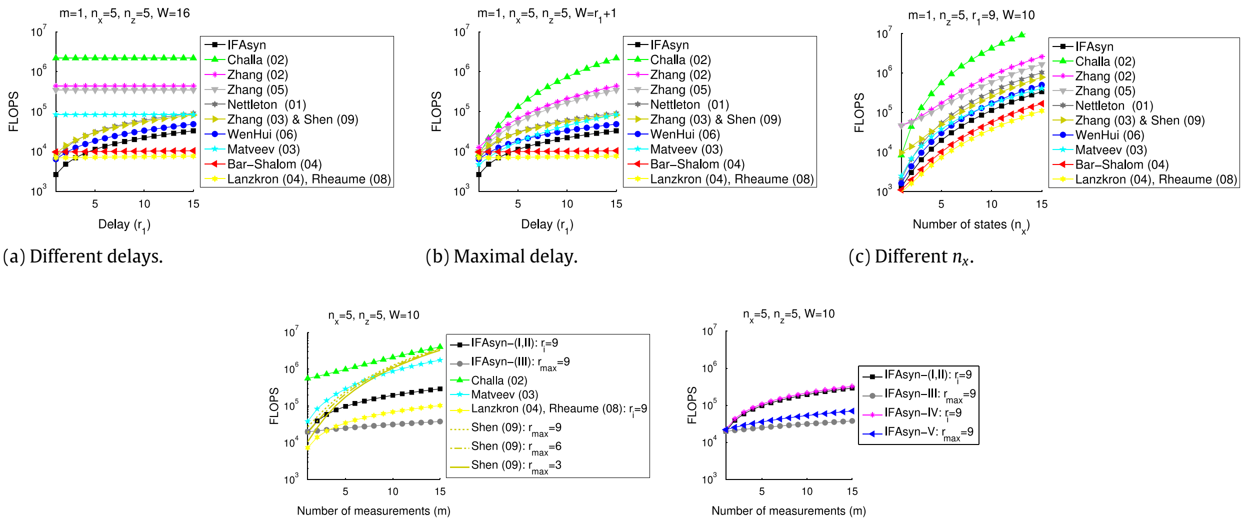
[6] We do not exactly compute the number of FLOPs in Shen et al. (2009) because the filter is too complex. Furthermore, we do not consider the extra operations to update the intermediate $(\hat{\boldsymbol{x}}_{j|j}, \boldsymbol{P}_{j|j})$. Hence, the number of FLOPs in Shen et al. (2009) in Table 3 is a lower bound. Further, under the *initial* assumptions and when $\boldsymbol{m} = 1$ the exact number of FLOPs in Shen et al. (2009) equal the ones in Zhang et al. (2003).

Table 2
Memory Comparison. The first column contains algorithms; the second, stored variables (following the notation of this paper for those which have the same meaning as ours, and the notation of their own papers for the rest); the third, memory requirements considering that $\boldsymbol{x}_t$ and $\boldsymbol{u}_{t,t_P}$ have n_x elements, and the fourth, a rated comparison (best less memory) considering $W > 2$.

ALG	Stored variables (for the last W time-stamps)	Memory needs	Rated
Shen et al. (2009) & WenHui et al. (2006) & Zhang et al. (2003)-AI	$\hat{\boldsymbol{x}}_{j\|j}$, $\boldsymbol{P}_{j\|j}$ and $\boldsymbol{u}_{j_N,j}$ with $j = t+1-W : t$	$W(n_x + n_x^2) + Wn_x$	1st
Nettleton and Durrant-Whyte (2001)	$\hat{\boldsymbol{y}}_{j\|j}$, $\boldsymbol{Y}_{j\|j}$ and $\boldsymbol{u}_{j_N,j}$ with $j = t+1-W : t$	$W(n_x + n_x^2) + Wn_x$	1st
IFAsyn-(I, II, III)	$\hat{\boldsymbol{y}}_{j\|j_P}$, $\boldsymbol{Y}_{j\|j_P}$, $\boldsymbol{i}_j$, $\boldsymbol{I}_j$ and $\boldsymbol{u}_{j_N,j}$ with $j = t+1-W : t$	$W(2n_x + 2n_x^2) + Wn_x$	2nd
Zhang et al. (2005)-AI	$\{\bigcup_{n=t+1-W:t}\{\hat{\boldsymbol{x}}_{n\|n}, \boldsymbol{P}_{n\|n}, \boldsymbol{y}_t^{(n)}, \boldsymbol{U}_t^{(n)}, \boldsymbol{B}_t^{(n)}, \boldsymbol{u}_{n,n-1}\}\}$	$W(2n_x + 3n_x^2) + Wn_x$	3rd
Zhang et al. (2002)-AI	$\{\bigcup_{n=t+1-W:t}\{\hat{\boldsymbol{x}}_{n\|n}, \boldsymbol{P}_{n\|n}, \hat{\boldsymbol{x}}_{t\|n}^{(n)}, \boldsymbol{P}_{t\|n}^{(n)}, \boldsymbol{T}_t^{(n)}, \boldsymbol{W}_t^{(n)}, \boldsymbol{u}_{n,n-1}\}\}$	$W(2n_x + 4n_x^2) + Wn_x$	4th
Challa et al. (2002) & Matveev and Savkin (2003)	Augmented $\boldsymbol{X}$ and $\boldsymbol{P}$ (with Wn_x and $(Wn_x)^2$ elements)	$Wn_x + W^2 n_x^2$	5th
Bar-Shalom et al. (2004) & Lanzkron and Bar-Shalom (2004), Rheaume and Benaskeur (2008)	$\hat{\boldsymbol{x}}_{j\|j}$, $\boldsymbol{P}_{j\|j}$ and $\boldsymbol{u}_{j_N,j}$ with $j = t+1-W : t$	$W(n_x + n_x^2) + Wn_x$	1st

Table 3
Computational Cost Comparison. First column, algorithms; second, number of FLOPs for carrying out a prediction and the assimilation of m measurements. Each $\boldsymbol{\xi}_{s_i,k_i,t}$ has n_z elements and is delayed r_i time-steps, $\boldsymbol{x}_t$ and $\boldsymbol{u}_{t_N,t}$ have n_x elements, the maximum permitted delay is $W-1$, $r_{max} = max(r_i)$, $B = \frac{7}{3}n_z^3 + 18n_z^2 + \frac{11}{3}n_z$ and $C = 4n_x^3 + n_x^2$.

ALG	Number of FLOPs
IFAsyn-(I, II)	$\{(\frac{26}{3}n_x^3 + 42n_x^2 + \frac{19}{3}n_x)\sum_i r_i\} + \{-\frac{19}{3}n_x^3 + (2n_z - 21)n_x^2 + (2n_z^2 + n_z - \frac{8}{3})n_x + B\}m + C$
IFAsyn-III	$(\frac{26}{3}n_x^3 + 42n_x^2 + \frac{19}{3}n_x)r_{max} + (-\frac{7}{3}n_x^3 + (2n_z m - 20)n_x^2 + (2n_z^2 m + n_z m - \frac{8}{3})n_x + Bm)$
Challa et al. (2002)	$(4n_x^3)W^3 + ((4n_z m + 1)n_x^2)W^2 + ((6n_z^2 m^2 + n_z m)n_x)W + (\frac{7}{3}n_z^3 m^3 + 18n_z^2 m^2 + \frac{11}{3}n_z m)$
Matveev and Savkin (2003)	$(2n_z n_x^2 m)W^2 + (4n_x^3 + 2((m^2+m)n_z - 1)n_x^2 + (2n_z^2 - n_z)m^2 n_x)W + ((2n_z m^2 + 3)n_x^2 + ((2n_z^2 - n_z)m^2 + 2n_z m)n_x + (B - n_z^2)m^2 + n_z^2 m)$
Zhang et al. (2002)-AI	$\{(7n_x^3 + (3n_z - 3)n_x^2 + (n_z^2 + n_z + \frac{1}{2})n_x + (\frac{7}{6}n_z^3 + 9n_z^2 + \frac{4}{3}n_z))W^2 + (-7n_x^3 + (n_z + 2)n_x^2 + (n_z^2 + 2n_z - \frac{3}{2})n_x + (\frac{7}{6}n_z^3 + 9n_z^2 + \frac{7}{3}n_z))W + (28n_x^3 + (6n_z - 6)n_x^2 + (6n_z^2 - 1)n_x + (B - n_z))\}m + C$
Zhang et al. (2005)-AI	$\{(4n_x^3 + (3n_z - 2)n_x^2 + n_z^2 n_x + \frac{1}{2}B)W^2 + (-4n_x^3 + (n_z + 1)n_x^2 + (n_z^2 + n_z - 1)n_x + \frac{1}{2}B)W + (12n_x^3 + (6n_z - 2)n_x^2 + 6n_z^2 n_x + B)\}m + C$
Nettleton and Durrant-Whyte (2001)	$\{(\frac{83}{3}n_x^3 + 98n_x^2 + \frac{55}{3}n_x)\sum_i r_i\} + \{-\frac{19}{3}n_x^3 + (2n_z - 17)n_x^2 + (2n_z^2 + n_z - \frac{14}{3})n_x + B\}m + C$
Zhang et al. (2003)-AI	$\{(\frac{61}{3}n_x^3 + (6n_z + 16)n_x^2 + (6n_z^2 + \frac{14}{3})n_x + B)\sum_i r_i\} + \{\frac{20}{3}n_x^3 + (2n_z + 39)n_x^2 + (2n_z^2 + n_z + \frac{16}{3})n_x + B\}m + C$
WenHui et al. (2006)	$\{(11n_x^3 + 63n_x^2 + 10n_x)\sum_i r_i\} + \{\frac{14}{3}n_x^3 + (2n_z + 40)n_x^2 + (2n_z^2 + n_z + \frac{16}{3})n_x + B\}m + C$
Shen et al. (2009) (see footnote 5)	$\{(2m^2 + 6m + \frac{31}{3})n_x^3 + (17 - m)n_x^2 + \frac{11}{3}n_x\}r_{max} + \{(4m^2 - 2m)n_x^3 + (2n_z(m^3 + m^2 + m) - m^2 + 3m + 3)n_x^2 + (n_z^2(2m^3 + 4m^2) + n_z(m^2 - m))n_x + (\frac{7}{3}n_z^3 m^3 + 17n_z^2 m^2 + \frac{11}{3}n_z m)\}$
Bar-Shalom et al. (2004)	$\{(2n_x^2)\sum_i r_i\} + \{\frac{107}{3}n_x^3 + (94 + 6n_z)n_x^2 + (\frac{55}{3} + 6n_z^2)n_x + B\}m + C$
Lanzkron and Bar-Shalom (2004), Rheaume and Benaskeur (2008)	$\{(2n_x^2)\sum_i r_i\} + \{\frac{59}{3}n_x^3 + (102 + 2n_z)n_x^2 + (2n_z^2 + n_z + \frac{46}{3})n_x + B\}m + C$
Validation	$\{\frac{7}{3}n_x^3 + (2n_z + 20)n_x^2 + (2n_z^2 + n_z + \frac{8}{3})n_x + (\frac{7}{3}n_z^3 + 20n_z^2 + \frac{17}{3}n_z - 1)\}m$

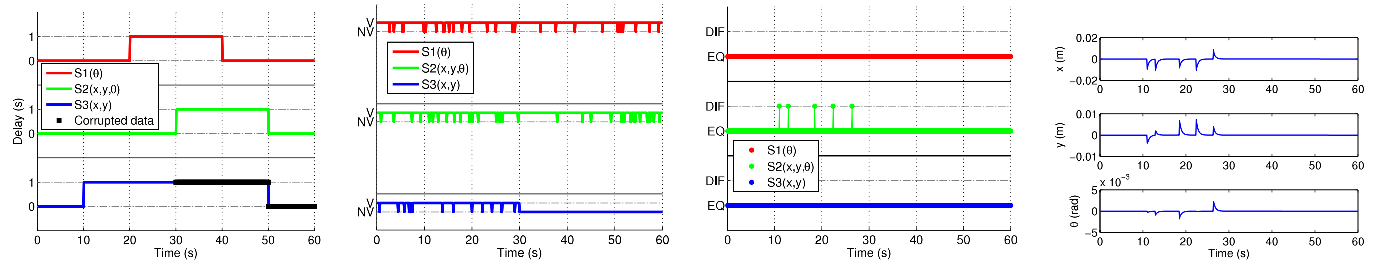


(a) Different delays. (b) Maximal delay. (c) Different n_x.

(d) Different m. (e) Validation.

Fig. 8. FLOPs comparison. Note that only the name of the first author and last two digits of the publication year are presented in the graphics to shorten the references.

and Bar-Shalom et al. (2004), Lanzkron and Bar-Shalom (2004), Nettleton and Durrant-Whyte (2001), Rheaume and Benaskeur (2008), WenHui et al. (2006), Zhang et al. (2003) and Zhang et al. (2002, 2005) assimilate each $\boldsymbol{\xi}_{s_i,k_i,t}$ separately, but while IFAsyn-(I, II) and Bar-Shalom et al. (2004), Lanzkron and Bar-Shalom (2004), Nettleton and Durrant-Whyte (2001), Rheaume and Benaskeur (2008), WenHui et al. (2006) and Zhang et al. (2003) have a linear dependency on the sum of the r_i and on m, Zhang et al. (2002, 2005) depend on $W^2 m$. IFAsyn-III and Challa et al. (2002), Matveev and Savkin (2003) and Shen et al. (2009) assimilate the set of variables in a single iteration: IFAsyn-III depends on the maximum delay (r_{max}) and on m, Challa et al. (2002) on W^3 and m^3, Matveev and



(a) Problem description. (b) Validation with delays. (c) Validation differences. (d) Influence on the state.

Fig. 9. Influence of the validation step on IFAsyn. V stands for valid, NV for non-valid, EQ for equal, and DIF for different.

Table 4
Linear system: (x_t, y_t) represents the mobile position, θ_t its orientation, and $(\Delta x_{t+1,t}, \Delta y_{t+1,t}, \Delta\theta_{t+1,t})^T$ the control signal.

	Transition	Sensor 1 (S1)	Sensor 2 (S2)	Sensor 3 (S3)
Model	$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{pmatrix} = \begin{pmatrix} x_t \\ y_t \\ \theta_t \end{pmatrix} + \begin{pmatrix} \Delta x_{t+1,t} \\ \Delta y_{t+1,t} \\ \Delta\theta_{t+1,t} \end{pmatrix} + \boldsymbol{v}_{t+1,t}$	$\boldsymbol{z}_{1,t+1} = (\theta_{t+1}) + \boldsymbol{v}_{1,t}$	$\boldsymbol{z}_{2,t+1} = \begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{pmatrix} + \boldsymbol{v}_{2,t}$	$\boldsymbol{z}_{3,t+1} = \begin{pmatrix} x_{t+1} \\ y_{t+1} \end{pmatrix} + \boldsymbol{v}_{3,t}$
Variance	$\boldsymbol{Q}_{t+1,t} = \begin{pmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & \frac{\pi}{180} \end{pmatrix}^2$	$\boldsymbol{R}_{1,t+1} = \left(\frac{\pi}{180}\right)^2$	$\boldsymbol{R}_{2,t+1} = \begin{pmatrix} 0.1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & \frac{2\pi}{180} \end{pmatrix}^2$	$\boldsymbol{R}_{3,t+1} = \begin{pmatrix} 0.05 & 0 \\ 0 & 0.05 \end{pmatrix}^2$

Savkin (2003) on W^2m and on Wm^2, and Shen et al. (2009) on $r_{max}m^2$ and on m^3. The asymptotic behavior of their number of FLOPs is shown in Fig. 8(a)–(c) when $m = 1$ and Fig. 8(d) when we have several $\xi_{s_i,k_i,t}$ to assimilate at t, all delayed $r_i = 9$. In Fig. 8(a) we increment its r_1 for a fixed W, in Fig. 8(b) we modify the maximum delay ($r_1 = W - 1$), in Fig. 8(c) we change the size of the state, and in Fig. 8(d) we modify m. Fig. 8(a)–(c) show that when $m = 1$ IFAsyn-(I, II, III) coincide and are the best optimal filters, followed always by WenHui et al. (2006) and sometimes by Matveev and Savkin (2003). Eventually, Matveev and Savkin (2003) can outperform our filters, however its dependency on W^2 instead of on r_1, makes us consider ours as computationally better for systems which can have big random delays. Besides, when the delay grows, IFAsyn-(I, II, III) are outperformed by the suboptimal (Bar-Shalom et al., 2004; Lanzkron & Bar-Shalom, 2004; Rheaume & Benaskeur, 2008), the best being Lanzkron and Bar-Shalom (2004) and Rheaume and Benaskeur (2008). Fig. 8(d) shows that IFAsyn-III is the best of the optimal algorithms when m grows, followed by IFAsyn-(I, II). The suboptimal (Lanzkron & Bar-Shalom, 2004; Rheaume & Benaskeur, 2008) is initially better, but as it assimilates $\xi_{s_i,k_i,t}$ one by one, it gets worse than IFAsyn-III. We also represent the behavior of Shen et al. (2009) at lower r_{max}, because when we drop the last *initial* assumptions, Shen et al. (2009) can potentially have less steps than IFAsyn-III, and we want to see what happen in that situation: Shen et al. (2009) degrades rapidly and IFAsyn-III is better with only a few $\xi_{s_i,k_i,t}$. The other filters are not presented in Fig. 8(d), because IFAsyn-(I, II) lower bound (Nettleton & Durrant-Whyte, 2001; WenHui et al., 2006; Zhang et al., 2002, 2003, 2005), and Lanzkron and Bar-Shalom (2004) and Rheaume and Benaskeur (2008) do the same for Bar-Shalom et al. (2004).

The previous analysis does not consider the validation step, whose number of FLOPs for m $\xi_{s,t,a}$ is shown in the last row of Table 3. The number of FLOPs in IFAsyn-IV and IFAsyn-V can be obtained incrementing the number of FLOPs in IFAsyn-II and IFAsyn-III with that value. Fig. 8(e) compares the number of FLOPs in our 5 filters as the number of $\xi_{s,t,a}$ grows. Although visually comparing IFAsyn-II with IFAsyn-IV, and IFAsyn-III with IFAsyn-V could suggest that the computational overload for the same number of $\xi_{s,t,a}$ is different, it is the same.

In conclusion, the memory and computation comparison show that when assimilating single $\xi_{s,k,t}$ the best *optimal* filters are WenHui et al. (2006) and any IFAsyn-(I, II, III). WenHui et al. (2006) minimizes the memory while only incrementing the computational cost of our filters and Matveev and Savkin (2003). Our filters usually minimize the computation needs of the rest, doubling the memory of WenHui et al. (2006). When several $\xi_{s,k,t}$ are assimilated together IFAsyn-III is usually the best computational option and Shen et al. (2009) the best memory-wise option. Whereas, the quickest suboptimal are Lanzkron and Bar-Shalom (2004) and Rheaume and Benaskeur (2008).

Finally, the filters Lanzkron and Bar-Shalom (2004), Nettleton and Durrant-Whyte (2001), Rheaume and Benaskeur (2008), WenHui et al. (2006) and Zhang et al. (2003) belonging to *group* 4 could also benefit from the use of managed lists to store their 3 variables to make them work in a similar fashion as IFAsyn-III. That is, they can be modified to have an equivalent memory and computational benefit.

5. Experimental results

In this section we illustrate with an example the usual influence of the OOS data in IFAsyn-(IV, V), the versions of IFAsyn that become suboptimal due to the inclusion of the validation step. IFAsyn-(I, II, III) are not considered because their optimality is ensured by their design. The example consists of comparing the results of IFAsyn-(IV, V) and KF for the mobile system in Table 4, with its 3 sensors (S1-compass, S2-sonars, S3-GPS) providing $\xi_{s,k,t}$ every 0.1 s during 60 s (600 $\xi_{s,k,t}$ per sensor). Both filters work with the same measurements, the KF with all of them without delay and IFAsyn-(IV, V) with a part of them delayed according to Fig. 9(a), which shows the delay (y-axis, 1 s means 10 time-steps) of the $\xi_{s,k,t}$ taken at each k (x-axis). In both cases S3 provides corrupted $\xi_{s,k,t}$ during the last 30 s and the filters have a validation test that should consider incorrect less than 5% of the non-corrupted $\xi_{s,k,t}$ ($\alpha = 0.05$). The test shows the behavior we have observed in our experiments with properly tuned filters: their results are only punctually different. Therefore, the influence of the OOS in the validation step is negligible.

The results of the experiment are presented in Fig. 9(b)–(d). Fig. 9(b) shows the results of the validation tests carried out by IFAsyn-(IV, V) with the delays in Fig. 9(a). The validation test rejects (NV) all the corrupted $\boldsymbol{\xi}_{s,k,t}$ from S3 (last 30 s), plus a small percentage of the rest. Although a similar figure could be presented for KF when $\boldsymbol{\xi}_{s,k,t}$ are not delayed, we prefer to include Fig. 9(c) that shows the differences in the validation tests of the two filters. We can observe that all $\boldsymbol{\xi}_{s,k,t}$ except 5 from S2 maintain the same validation results. This small difference, only a 0.28% of all the $\boldsymbol{\xi}_{s,k,t}$, is due to the presence of delays in S1 and S3 between the period of 10–30 s, which makes IFAsyn-(IV, V) validate $\boldsymbol{\xi}_{s,k,t}$ from S2 before a part of the $\boldsymbol{\xi}_{s,k,t}$ from S1 and S3 have been assimilated. The influence in the values of the states is shown in Fig. 9(d), with 5 abrupt changes in all the states for the same instant of times. Considering that during the experiments the value of x, y and θ vary 60, 25 m and 1 rad, the biggest change in the three axis is around 0.02% of the total change. Thus, the differences are negligible, both in the number of $\boldsymbol{\xi}_{s,k,t}$ whose validation test is different and in the changes in the values of the states.

The experiment shows the usual influence of the validation test in IFAsyn-(IV, V) for a well-tuned case. When the models or the tests are badly tuned, both the KF without delayed $\boldsymbol{\xi}_{s,k,t}$ or IFAsyn-(IV, V) with OOS data can obtain erroneous results. That is, a correct validation test and parameter tuning is needed by all the filters, and although the OOS data make IFAsyn-(IV, V) lose its optimality, they do not necessarily aggravate the problems that the validation test imposes on KF.

6. Conclusions

We present several filters to estimate the state of linear control systems for the OOSP with corrupted data. IFAsyn-(I, II, III) are optimal solutions that avoid recalculating the information of the already received measurements, although they re-propagate it. They are computationally better than other optimal solutions and easier to implement as they consist in a cleverly organized group of KF prediction, IF update operations, and state projections. IFAsyn-(IV, V) extend IFAsyn-(II, III) with the inclusion of a validation step that lets them detect and reject erroneous measurements. Our filters are also more general (see Table 1 for a complete comparison).

References

Anxi, Y., Diannong, L., Weidong, H., & Zhen, D. (2005). A unified out-of-sequence measurements filter. In *Proc. of the 2005 IEEE international radar conference, May* (pp. 453–458).

Bar-Shalom, Y. (2002). Update with out of sequence measurements in tracking: exact solution. *IEEE Transactions on Aerospace and Electronics Systems*, *38*(3), 769–778.

Bar-Shalom, Y., Mallick, M., Chen, H., & Washburn, R. (2004). One step solution for the general out of sequence measurement problem in tracking. *IEEE Transactions on Aerospace and Electronics Systems*, *40*(1), 27–37.

Besada-Portas, E. (2004). Fusion Multisensorial de medidas temporalmente desordenadas: aplicacion a robots autonomos mobiles. Ph.D. thesis. Universidad Complutense Madrid.

Besada-Portas, E., Lopez-Orozco, J. A., Besada, J., & de la Cruz, J. M. (2009). Multisensor out of sequence data fusion for estimating the state of discrete control systems. *IEEE Transactions on Automatic Control*, *54*(7), 1728–1732.

Besada-Portas, E., Lopez-Orozco, J. A., & de la Cruz, J. M. (2007). Multisensor out-of-sequence data fusion for estimating the state of dynamic systems. In *Proc. of information, decision and control (IDC 2007), February* (pp. 348–353).

Challa, S., Evans, R. J., Wang, X., & Leggy, J. (2002). A fixed lag-smoothing solution to out-of-sequence information fusion problems. *Communication in Information Systems*, *2*(4), 325–348.

Cowell, R., Dawid, P., Lauritzen, S., & Spiegelhalter, D. (1999). *Networks and expert systems*. Springer.

Feng, X., Ge, Q., & Wen, C. (2008). Optimal update with one step out-of-sequence measurements for wireless multisensor network. In *International conference on wavelet analysis and pattern recognition, Vol. 2, August* (pp. 826–831).

Fukunaga, K. (1990). *Introduction to statistical pattern recognition*. Academic Press.

Hall, D. L. (1992). *Mathematical techniques in multisensor data fusion*. Artech House.

Hespanha, J. P., Naghhtabrizi, P., & Xu, Y. (2007). A survey of recent results in networked control systems. In *Proc. of IEEE, Vol. 95, January*.

Hilton, R. D., Martin, D. A., & Blair, W. D. (1993). Tracking with time-delayed data in multisensor systems. In *AD-A355269, August*.

Ito, M., Tsujimichi, S., & Kosuge, Y. (1998). Target tracking with time-delayed data in multiple radar system. In *Proc. of the 37th SICE annual conference. International session papers, July*.

Johnson, R. A, & Wichern, D. W. (1998). *Applied multivariate statistical analysis*. Prentice Hall.

Kosaka, A., Meng, M., & Kak, A. C. (1993). Vision-guided mobile robot navigation using retroactive updating of position uncertainty. In *Proc. of the 1993 IEEE international conference on robotics and automation, Vol. 2, May* (pp. 1–7).

Lanzkron, P. J., & Bar-Shalom, Y. (2004). A two-step method for out-of-sequence measurements. In *Proc. of 2004 IEEE aerospace conference, Vol. 3, March*.

Lauritzen, S. L. (1992). Propagation of probabilities, means and variances in mixed graphical models. *Journal of the American Statistical Association*, *87*, 1098–1108.

Lopez-Orozco, J. A., de la Cruz, J. M., Besada, E., & Rupiezed, P. (2000). An asynchronous robust and distributed multisensor fusion system for mobile robots. *The International Journal of Robotics Research*, *19*(10), 914–932.

Lu, X., Zhang, H., Wang, W., & Teo, K. L. (2005). Kalman filtering for multiple time delay measurements. *Automatica*, *41*(8), 1455–1461.

Mallick, M., Coraluppi, S., & Carthel, C. (2001). Advances in asynchronous and decentralized estimation. In *Proc. of IEEE aerospace conference, March* (pp. 1873–1888).

Matveev, A., & Savkin, A. V. (2003). The problem of state estimation via asynchronous channels with irregular transmission times. *IEEE Transactions on Automatic Control*, *48*(4), 670–676.

Murphy, K. P. (1998). Filtering, smoothing and the junction tree algorithm. Technical report. Universidad de California at Berkeley, Department of CS.

Mutambara, G. O. (1998). *Decentralized estimation and control for multisensor fusion*. CRC Press LLC.

Nettleton, E. W., & Durrant-Whyte, H. (2001). Delayed and asequent data in decentralized sensing networks. In *Proc. of SPIE conf. n 4571* (pp. 1–9).

Rheaume, F., & Benaskeur, A. R. (2008). Forward prediction-based approach to target-tracking with out-of-sequence measurements. In *47th IEEE conference on decision and control, December* (pp. 1326–1333).

Shen, X., Zhu, Y., Song, E., & Luo, Y. (2009). Optimal centralized update with multiple local out-of-sequence measurements. *IEEE Transactions on Signal Processing*, *57*(4), 1551–1562.

Sinopoli, B., Schenato, L., Franceschetti, M., Poolla, K., Jordan, M. I., & Sastry, S. S. (2004). Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control*, *49*(9), 1453–1464.

WenHui, Z., Lin, L., GuoHai, C., & AnXi, Y. (2006). Optimal updater with multistep out of sequence measurements in target tracking. In *Proc. of the 8th international conf. on signal processing, Vol. 4* (pp. 281–284).

Zhang, K., Li, X. R., & Chen, Y. (2002). Optimal update with out of sequence measurements for distributed filtering. In *Proc. of 5th international conference of information fusion* (pp. 1519–1526).

Zhang, K., Li, X., & Chen, Y. (2003). Multi-sensor multi-tracking with out-of-sequence measurements. In *Proc. of ISIF 2003* (pp. 672–679).

Zhang, K., Li, X. R., & Zhu, Y. (2005). Optimal update with out-of-sequence measurements. *IEEE Transactions on Signal Processing*, *53*(June), 1992–2004.

Zhang, H., Xie, L., Zhang, D., & Soh, Y. C. (2004). A reorganized innovation approach to linear estimation. *IEEE Transactions on Automatic Control*, *49*(10), 1810–1814.

Eva Besada-Portas received the M.Sc. degree in physics and the Ph.D. degree in computer engineering from the Complutense University of Madrid (UCM), Spain in 1997 and 2004, respectively. In 2002, she joined the Department of Computer Architecture and Automatic Control, UCM, as a Teaching Assistant. Since 2005 she has been an Assistant professor at UCM. Simultaneously, and since 2006, she has been a Postdoctoral Visiting Researcher with the Department of Computer Science, University of New Mexico, Albuquerque, USA. Her current research interests include optimal control, evolutionary algorithms, multisensor fusion systems, and machine-learning techniques.

Jose A. Lopez-Orozco received the M.Sc. and Ph.D. degrees in physics from the Physics Science Faculty, Complutense University of Madrid (UCM), Madrid, Spain, in 1991 and 1999, respectively. In 1992, he joined the Department of Computer Architecture and Automatic Control, UCM, as a Teaching Assistant and, later as an Assistant Professor. Since 2002, he has been an Associated Professor. His current research interests include applications of Automatic Control, artificial neural networks, robotics, and multisensor fusion systems.

Juan Besada is an Associate Professor in Universidad Politécnica de Madrid. He received his Ph.D from the same University in 2001. He has been working with GPDSCEDITECSSR department from 1995. His main interests are Air Traffic Control and Management, Data Fusion, Pattern Analysis, and Localization Technologies. He has been working in many national and international projects, in cooperation with industrial partners, such as INDRA, AENA, EUROCONTROL, Boeing R&TE,… on the application of new information fusion, optimization, and pattern recognition technologies to their operational systems.

Jesus M. de la Cruz received the M.Sc. and Ph.D. degrees in physics from the Physics Science Faculty, Complutense University of Madrid (UCM), Madrid, Spain, in 1979 and 1984, respectively. From 1985 to 1990, he held teaching appointments at the UCM and at the Open University of Spain (UNED), Madrid. From 1990 to 1992, he was with the Department of Electronics, Cantabria University of Santander, Santander, Spain. In October 1992, he joined the Department of Computer Architecture and Automatic Control, UCM, where he was the Dean from 1997 to 2001 and is currently a Full Professor and the Head of the Automatic Control and Robotics Group. His current research interests include broad aspects of automatic control and its applications, real-time control, optimization, statistical learning, and robotics.