DISTRIBUTED SENSOR FUSION FOR AIDED INERTIAL NAVIGATION SYSTEMS

Cassio G. Lopes and Jacques Waldmann

Instituto Tecnologico de Aeronautica - Dept. of Electronics Engineering
Sao Jose dos Campos, SP - Brazil

ABSTRACT

We formulate distributed estimation protocols built on Kalman filter implementations to be embedded in sensor networks applications, namely those related to navigation problems. We have pursued protocols that (a) maximizes energy savings and (b) avoids the synchronization typically required to implement consensus protocols. Our approach leads to robust algorithms that draw a minimum of energy from the network, increasing autonomy, crucial in defense and critical applications. A pilot example validates the proposed algorithms, showing that the distributed setup is able to match the performance of a central node that fuses the estimates captured across the network.

Index Terms— Distributed estimation, data fusion, Kalman filtering, inertial navigation

1. INTRODUCTION

We propose simple yet robust distributed algorithms to perform data fusion in a network of nodes, capturing space-time data related to an event of interest taking place in the field [1, 2]. Besides typical sensor networks applications, after proper linearization and discretization, the algorithms presented in this work may also be employed for fusion of diverse sensors in aided inertial navigation problems. With the proper errors model, it also fits typical problems in the strapdown navigation context: initial alignment and sensors calibration. More specifically, applications for the proposed framework range from UAV autonomous operation to aircraft carrier navigation aiding, sounding rocket tracking by ground radar stations [3], as well as aided INS [4, 5].

By building on individual local Kalman filters (LKFs), which represent a lower performance bound, we establish simple fusion protocols that are able to outperform the LKF setup, also matching the performance of a centralized fusion strategy, employed as benchmark. Our goal is to arrive at distributed fusion protocols that are (a) fully distributed, (b) preserve network autonomy and (c) approach, as much as possible, the performance of a global Kalman solution, which sets the network performance goal.

The presented formulation and solutions employ general models, and they are evaluated via a simple yet representative 2D vehicle tracking problem, which conceptually captures all the relevant issues in distributed estimation implementations. Inspired by recent results on distributed adaptive filtering [7]–[10], we also start exploring the concept of Adaptive Networks in the context of inertial navigation.

This work was funded by the project FINEP/INPE/CTA 01.050770.00 Ref. FINEP 2769/2005 - Sistemas Inerciais para Aplicação Aeroespacial - SIA.

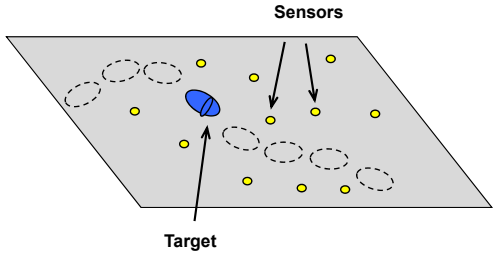

Fig. 1. Network of sensors.

2. THE RISE OF DISTRIBUTED ESTIMATION

What is a distributed algorithm? Different answers have been given in the literature, and the question is open to some dispute. More recently, the notion of a *fully* distributed algorithm arose in the sensor networks and control community. In the sense employed in this work, without discussing momentarily the need for such systems, an algorithm embedded in a network of sensors is considered fully distributed if any particular sensor/node relies solely on local information, and exchanges information, if or when necessary, only with its direct neighbors (i.e., peer-to-peer protocols, or simply "P2P"). Routing protocols are also possible, which may lead to distributed implementations as well, but will not be covered here.

2.1. Why distributed?

In several quotidian situations, space-time events take place in the field, and often prompt action is necessary. Average temperature estimation, gas leakage detection, pollution control, target tracking, source description and localization, to name a few, are among the emerging applications in sensor networks, as well as in the newborn field of adaptive networks [1, 2, 9].

By spreading sensors out in the field, the probability of event capture and coverage is increased, and since a great deal of processing is required to deal with the large amounts of data captured by the nodes, an efficient strategy for processing is required. It is well known that a node spends an order of magnitude more energy and communication resources to send a bit for processing to a central node, whose result must be transmitted afterwards back to the node, than if the bit is locally processed, which is known in the literature as "in-network" processing [1]. In other words, distributed processing means network autonomy. Besides, the need of a powerful processor, located at the central node, in charge of the processing tasks, is avoided, and robustness is achieved as a byproduct.

In sounding rocket tracking and aided INS for guidance and navigation, a suit of sensors capture motion-related data so that position,

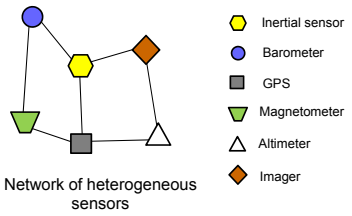

Fig. 2. Heterogeneous network of sensors.

velocity and attitude can be accurately estimated, as well as the sensor errors to perform proper calibration of the instruments. Integration of the various sensors by distributing the computational burden among the sensor nodes is highly desirable for the aforementioned reasons.

2.2. Cooperation and adaptivity

Each event leaves a space-time signature in the field that must be captured by the network. It is typically time-varying, which suggests that another desirable feature to embed in the nodes is the ability to learn from the captured data, or adaptivity [1, 9, 10]. Several learning rules are possible [9, 10], and when the underlying process of interest may be modeled as a Markov process, the Kalman filter is particularly handy.

The distributed solution, to a certain extent, can be understood as a factorization of the global optimization problem into subproblems, placed at the nodes, with each node attempting to solve a small fraction of the general problem. However, it becomes necessary to aggregate these individual solutions, obtained with limited local data, in order to recover, or to approach, as much as possible, the performance of the original global solution, running in a central node aware of everything.

This aggregation procedure is generally referred to as *data fusion*, and implies cooperation, directly or indirectly, among the nodes. Thus, cooperation is just general terminology to some sort of data fusion procedure that takes place at the node level, throughout the distributed network.

3. PROBLEM FORMULATION

In its complete formulation, the problem of inertial navigation involves the solution of non-linear and time-varying equations. Upon linearization and discretization, typical estimation problems that arise in navigation applications may be well captured by Markov models. Examples of such problems are initial alignment and calibration of inertial sensors in strapdown solutions by means of integration with auxiliary sensors, such as magnetometers, GPS receiver, altimeter, etc, yielding improved accuracy (see Fig. 2).

We proceed by assuming that the process of interest follows a state-space model of the form

$$x_{i+1} = F_i x_i + G_i u_i \tag{1}$$

Where F_i and G_i are $M \times M$ known matrices, so that the state dimension is $M < N$, and u_i is an $M \times 1$ white process with covariance matrix Q_i. The state of the process is only indirectly observed by a network of N sensors via a linear measurement model. Node k captures measurements about the process x_i at time i via

$$y_{k,i} = H_{k,i} x_i + v_{k,i} \tag{2}$$

where $v_{k,i}$ is a white sequence with covariance matrix $R_{k,i}$ which models measurement noise, and $H_{k,i}$ is a node-dependent local observation matrix, which accounts for heterogeneous sensors. Conceptually, the goal is to reconstruct the process trajectory $\{x_j\}_{j=0,\ldots,i}$ from the measurements $\{y_{\ell,j}\}_{j=0,\ldots,i}$ collected at the $\ell = 1, \ldots, N$ nodes.

3.1. Global Kalman filter (GKF)

A natural tool to tackle such problem is the Kalman filter (KF). In a typical solution to the problem, all the measurements $y_{k,i}$ collected throughout the network are stacked onto a global column vector

$$y_i \triangleq \begin{bmatrix} y_{1,i} \\ y_{2,i} \\ \vdots \\ y_{N,i} \end{bmatrix} \tag{3}$$

resulting in the global quantities

$$H_i \triangleq \mathrm{col}\{H_{1,i}, H_{2,i}, \ldots, H_{N,i}\} \tag{4}$$

and

$$R_i \triangleq \mathrm{diag}\{R_{1,i}, R_{2,i}, \ldots, R_{N,i}\} \tag{5}$$

The global vector (3) is then presented to a large global Kalman filter (GKF), located at a hypothetical central node, whose equations in the prediction form are given by [6]

$$\begin{aligned} e_i &= y_i - H_i \hat{x}_i & (6)\\ R_{e,i} &= H_i P_i H_i^* + R_i & (7)\\ K_{p,i} &= F_i P_i H_i^* R_{e,i}^{-1} & (8)\\ P_{i+1} &= F_i P_i F_i^* + G_i Q_i G_i^* - K_{p,i} R_{e,i} K_{p,i}^* & (9)\\ \hat{x}_{i+1} &= F_i \hat{x}_i + K_{p,i} e_i + G_i \overline{u}_i & (10) \end{aligned}$$

In the above equations, $\hat{x}_i \triangleq \hat{x}_{i|i-1}$ is the process estimate delivered by the GKF, and it considers all the observations gathered across the network up to time $i-1$. The quantity e_i is known as the process innovation, whose covariance matrix is given by $R_{e,i}$. Defining the state error vector as $\tilde{x}_i = x_i - \hat{x}_{i|i-1}$, then follows its corresponding covariance matrix $P_i \triangleq E\tilde{x}_i\tilde{x}_i^*$, and the prediction Kalman gain $K_{p,i}$. The vector $\overline{u}_i = Eu_i$ captures the input mean trajectory.

There are several problems with the GKF solution (6)–(10), besides those mentioned earlier. One of the ultimate problems is the need to invert large matrices (e.g., $R_{e,i}$), potentially giving rise to numerical instabilities. Another is the need for a powerful processor to handle such computations. On the other hand, the GKF has a strong conceptual appeal, in the sense that it can be considered as a lower bound on the state mean-square error (MSE). Any distributed solution should aim at this bound.

3.2. Local Kalman filters (LKF)

At the other end of the solutions' spectrum, is the LKF, in which stand-alone, non-cooperative KFs are run at the nodes. As matter of fact, isolated KFs do not actually represent a solution, since they are not aimed at the global solution. However, they may be viewed as the building blocks, over which cooperative protocols may be implemented in order to approach, as much as possible, the global solution. Therefore, they may also be considered as an MSE upper bound.

The optimal estimate delivered by the individual KF at node k, time i, considering all the local observations (i.e., $y_{k,i}$) up to time $i-1$ is denoted by $\hat{x}_{k,i|i-1} \triangleq \hat{x}_{k,i}$, and it is obtained, in the prediction form, via

$$
\begin{aligned}
e_{k,i} &= y_{k,i} - H_{k,i}\hat{x}_{k,i} && (11)\\
R_{e,k,i} &= H_{k,i}P_{k,i}H_{k,i}^* + R_{k,i} && (12)\\
K_{p,k,i} &= F_i P_{k,i} H_{k,i}^* R_{e,k,i}^{-1} && (13)\\
P_{k,i+1} &= F_i P_{k,i+1} F_i^* + G_i Q_i G_i^* - K_{p,k,i} R_{e,k,i} K_{p,k,i}^* && (14)\\
\hat{x}_{k,i+1} &= F_i \hat{x}_{k,i} + K_{p,k,i} e_{k,i} + G_i \overline{u}_i && (15)
\end{aligned}
$$

where now $P_{k,i} \triangleq E\tilde{x}_{k,i}\tilde{x}_{k,i}^*$ and $R_{e,k,i} = Ee_{k,i}e_{k,i}^*$.

4. DISTRIBUTED SENSOR FUSION

As previously suggested, LKFs will be employed as the building blocks of our distributed estimation algorithms. Each sensor in the network is equipped with a KF of the form (11–15), which can communicate with its nearby KFs, in compliance with the existing communication topology. On top of the existing individual KFs, we adopt cooperation protocols that perform data fusion in a distributed manner. More specifically, the KFs exchange their estimates $\hat{x}_{k,i}$ with their direct neighbors. Upon reception, the estimates from the neighborhood are aggregated, and a fused local estimate is generated, with better statistical properties. Different aggregation rules lead to different performances, in terms of state mean square error and computational complexity. Here we will explore a simple fusion technique, exposed in the sequel.

4.1. Distributed convex fusion

One simple way to achieve data fusion is to resort to locally-convex averaging procedures. In this scheme, the local KFs diffuse their estimates to their neighboring nodes across the network, following a *diffusion* mode of cooperation [9, 10]. As consequence, a particular node, say k, will receive estimates from its direct neighbors, according to the network topology. Upon reception of such estimates, a better estimate for x_i can be obtained via a simple local fusion rule:

$$
\overline{x}_{k,i} = \sum_{\ell \in \mathcal{N}_k} c_{k\ell}\hat{x}_{\ell,i}\ , \quad \sum_{\ell} c_{k\ell} = 1 \tag{16}
$$

where $\{c_{k\ell}\}$ is a set of *local* combiners that fuse the estimates $\{\hat{x}_{\ell,i}\}_{\ell\in\mathcal{N}_k}$, received from the neighborhood, into a better estimate $\overline{x}_{k,i}$. Here $\mathcal{N}_k$ stands for node k's neighborhood, which is the set of nodes directly connected to node k, including itself.

The rule (16) may be interpreted as a weighted least-squares fusion of the nearby estimates via the combiners $c_{k\ell}$, which have to be designed somehow. Such combiners may be topology-dependent only, or may also take into account the statistics of the received estimates in order to improve the fusion process.

Typical designs for the combiners c's that account only for network topology are given by the Metropolis, Laplacian and nearest neighbor rules [8, 10]. The Metropolis rule is defined as follows. Let n_k and n_ℓ denote the degree for nodes k and ℓ, i.e., $n_k = |\mathcal{N}_k|$, and choose

$$
\begin{cases}
c_{k\ell} = 1/\max(n_k, n_\ell) & \text{if } k \neq \ell \text{ are linked}\\
c_{k\ell} = 0 & \text{for } k \text{ and } \ell \textit{ not} \text{ linked}\\
c_{kk} = 1 - \sum_{\ell\in\mathcal{N}_k/k} c_{k\ell} & \text{for } k = \ell
\end{cases} \tag{17}
$$

Now, note that the set of all combiners in the network render a matrix $C = [c_{k\ell}]$ which provides information on the network topology: a nonzero entry $c_{k\ell}$ means nodes k and ℓ are connected. From C we may define the Laplacian rule:

$$
C = I_N - \kappa\mathcal{L} \tag{18}
$$

where $\mathcal{L} = \mathcal{D} - A_d$, with $\mathcal{D} = \text{diag}\{n_1, \ldots, n_N\}$, $\kappa = 1/n_{\max}$, where $n_{\max}$ is the maximum node degree, and A_d is the $N \times N$ network adjacent matrix, formed as

$$
[A_d]_{k\ell} = \begin{cases} 1 & \text{if } k \text{ and } \ell \text{ are linked}\\ 0 & \text{otherwise}\end{cases} \tag{19}
$$

By definition, a node is linked to itself, i.e., $[A_d]_{kk} = 1$. Last, but not least, in the nearest neighbor rule the combiners matrix C is defined as

$$
\begin{cases}
c_{k\ell} = \frac{1}{|\mathcal{N}_k|}\ , & \ell \in \mathcal{N}_k\\
c_{k\ell} = 0 & \text{otherwise}
\end{cases}
$$

Summarizing, our distributed algorithm comprises (a) Set up a network of KFs given by (11)–(15), (b) For time $i \geq 0$, and for every node k in the network, do

1. Exchange estimates among neighboring KFs
2. Implement local fusion (16)

The resulting quantity $\overline{x}_{k,i}$ may be regarded as a simple fusion of node k's neighboring estimates, with improved statistical properties, and can be readily used.

Another way to design the c's involves statistical information about the received neighboring estimates $\{\hat{x}_{\ell,i}\}_{\ell\in\mathcal{N}_k}$. For instance, a possible design accounting for the estimates statistics is [6]:

$$
c_{k\ell} = \frac{1}{a_k\sigma_\ell^2}\ , \quad a_k = \sum_{\ell\in\mathcal{N}_k}\frac{1}{\sigma_\ell^2} \tag{20}
$$

Here σ_ℓ^2 is a measure of the estimation error variance, which in this work is replaced, for the sake of simplicity, by the measurement error variance, assuming all LKF estimation errors converge. Intuitively, (20) says that if an estimate has a larger variance, then it should be assigned a smaller weight. For Gaussian signals and depending on how σ_ℓ^2 is computed, schemes of this kind can be interpreted as maximum likelihood fusion.

4.2. Centralized fusion

In this implementation, the network of LKFs report their local estimates at each iteration to a central node, in charge of fusing all the received estimates into a better estimate, which is then sent back to the nodes for local use. The fusion rule employed at the central node is given by

$$
\overline{x}_i^c = \sum_{k=1}^{N} c_k\hat{x}_{k,i}\ , \quad \sum_k c_k = 1 \tag{21}
$$

for a set of convex combiners c_k. This strategy, although feasible, clearly defeats the original purpose of a fully distributed implementation, and it will be used for benchmark only. Note that the role of the central node here is *only* to implement the convex fusion (21), and it is not to be confused with the operation of the global Kalman filter, presented in Section 3.1. In other words, the central node here *does not* run a global Kalman filter.

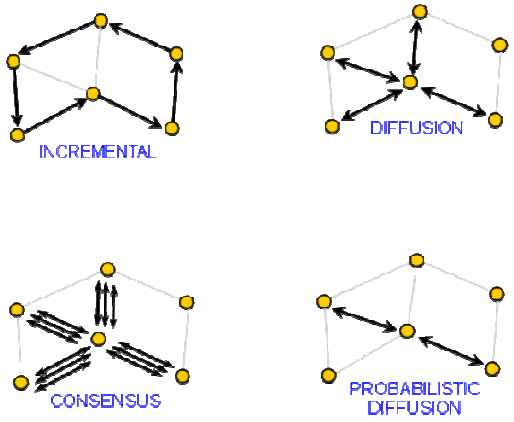

Fig. 3. Modes of cooperation in an adaptive network.

4.3. Adaptive networks: a perspective

Another strategy that we will glance at is based on recent theory developed in the context of distributed adaptive filtering: adaptive networks [7]–[10]. To begin with, as illustrated by Fig. 3, other modes of cooperation are possible, besides the *diffusion protocol* described in Section 4.1 and embedded in the local convex fusion rule (16). When operating in the *incremental mode* of cooperation, nodes circulate their estimates across a pre-selected cycle, whereby a node receives an estimate from its previous node in the cycle, updates it and pass it to the next node. In a *consensus mode*, nodes repeat the diffusion exchange of information a number of times, so that the averaging procedure is improved. The *probabilistic diffusion mode* is a relaxation of the full diffusion mode, in order to save energy and communication resources (refer to [9, 10] for details). On top of that, alternative learning rules could be used.

Another major difference in an adaptive network is that, in addition to the fusion step (16), the resulting aggregate estimate $\overline{x}_{k,i}$ is injected into the local learning rule, the Kalman update equation in our case. As a consequence, a possible "diffusion Kalman filter" could be

$$\overline{x}_{k,i} = \sum_{\ell \in \mathcal{N}_k} c_{k\ell} \hat{x}_{\ell,i} \ , \quad \sum_{\ell} c_{k\ell} = 1 \tag{22}$$

$$\overline{e}_{k,i} = y_{k,i} - H_{k,i}\overline{x}_{k,i} \tag{23}$$

$$R_{e,k,i} = H_{k,i}P_{k,i}H_{k,i}^* + R_{k,i} \tag{24}$$

$$K_{p,k,i} = F_i P_{k,i} H_{k,i}^* R_{e,k,i}^{-1} \tag{25}$$

$$P_{k,i+1} = F_i P_{k,i+1} F_i^* + G_i Q_i G_i^* - K_{p,k,i} R_{e,k,i} K_{p,k,i}^* \tag{26}$$

$$\hat{x}_{k,i+1} = F_i \overline{x}_{k,i} + K_{p,k,i}\overline{e}_{k,i} + G_i \overline{u}_i \tag{27}$$

Algorithm (22) is a first experiment inspired on the dramatic improvement experienced by networks that run adaptive protocols where the fused estimates are injected into the update "learning" equation. In the Kalman case, where the generated estimates are already mean-square optimal in some sense (for instance, locally, or globally), additional steps might be necessary in order to handle the correlation among estimates and drive performance closer to the GKF.

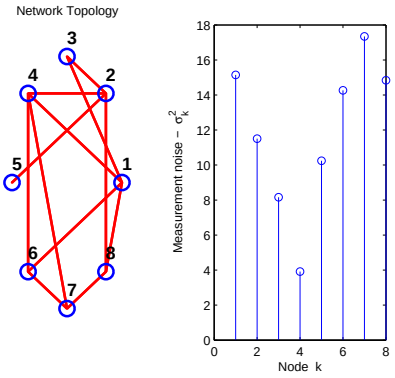

Fig. 4. Network settings. Left: Topology. Right: σ_k^2 distribution.

5. SIMULATION RESULTS

To evaluate the framework developed, we pose a vehicle tracking problem in the plane. Despite its simplicity, this problem captures the essence of the concepts involved in a typical distributed (adaptive) estimation scenario. Basically, the process of interest is implemented with

$$F = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} , \tag{28}$$

with $\theta = 4.8759^o$. Matrices G, Q and $H_{k,i}$ are identity. The noise covariance matrices are $R_{k,i} = \sigma_k^2 \cdot I$, where σ_k^2 is randomly chosen, and so is the network topology. The state estimates are initialized with zeros, and the state error covariance matrices are initialized with $\epsilon^{-1}I$, where ϵ is a small number (For instance, 10^{-3}).

We employ as performance bounds the individual LKF setup from Section 3.2 and the global Kalman solution (GKF) from Section 3.1. In this setup we compare the convex averaging distributed algorithm (Section 4.1) with the centrally-fused estimates (21), employed for benchmark. We also present, as a first experiment, the diffusion Kalman algorithm (Section 4.3).

Two curves are presented: the trajectories reconstructed via the algorithms presented here, and the state propagated mean square error (MSE). The former compares the true vehicle trajectory against the trajectory obtained by the distributed algorithms at a node picked randomly from the network. The latter employs a network-averaged MSE curve which captures the network average performance [9], avoiding tedious comparisons among all the existing nodes. The network average MSE is defined as

$$MSE(i) = \frac{1}{N}\sum_{k=1}^{N} E\|\tilde{x}_{k,i}\|^2 \tag{29}$$

Fig. 4 shows the settings for our example. The actual and estimated vehicle trajectories are depicted in Fig. 5, where the dotted trajectory represents the true vehicle path.

The MSE curves for the all the algorithms are plotted in Fig. 6. Note how the distributed algorithm proposed in Section 4.1 outperforms the individual LKFs, and even match the centrally-fused estimates. The ensemble average curves were generated with 1000 experiments. As expected, the global solution (GKF) is the performance "bound". The diffusion KF was outperformed by the pure distributed averaging procedures, however it was included to illustrate the potential of the method. If properly tuned, it is expected

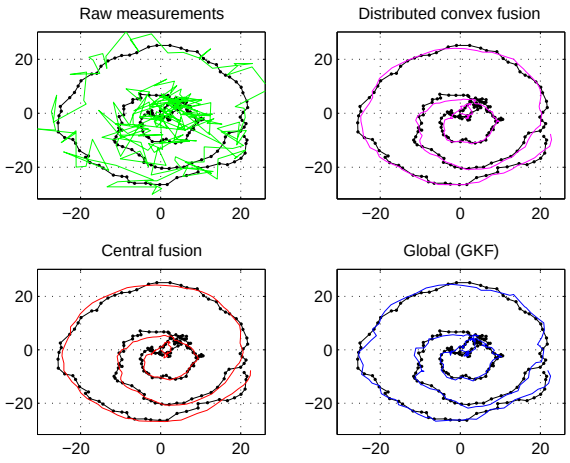

Fig. 5. Trajectories for the algorithms. Captured at node 2.

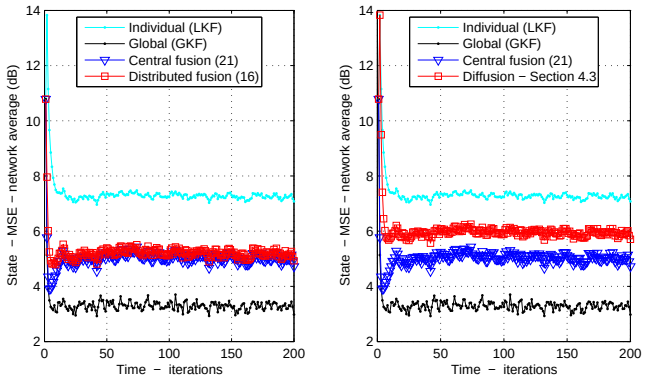

Fig. 6. State mean square error.

to lead to considerably better results, as it is the case in its original adaptive filtering scenario.

It is important to point out that these curves could be improved by implementing consensus protocols, but at a much higher communication and energy cost (See Fig. 3).

6. CONCLUDING REMARKS AND FUTURE WORK

Unlike existing consensus protocols [11], we have pursued protocols that (a) maximizes energy savings and (b) may avoid the synchronization typically required to implement those protocols. Our approach leads to quite robust algorithms that draw a minimum of energy from the network, increasing autonomy, crucial in defense and critical applications. The price paid is that the individual nodes' performance may not be so close to the global solution. However, the protocols presented in this work outperform the non-cooperative solution, and match the performance of the centralized fusion, as corroborated by simulations.

We are currently studying protocols that incorporate the state error covariance matrices $P_{k,i}$ in the fusion process, which is expected to further improve the estimates. Given two estimates x_1 and x_2, with covariance matrices $P1$ and $P2$, respectively, the optimal fusion rule in the mean-square sense is the rule

$$P^{-1}x = P_1^{-1}x_1 + P_2^{-1}x_2 \tag{30}$$

where x is the fused estimate, with corresponding covariance matrix P. There are efficient strategies to circumvent the need to know the global covariance matrix P, so that (30) can be easily generalized to several estimates and applied in the distributed setup proposed here [3].

We are also working to validate the results in more realistic scenarios, accounting for sensor non-observability, non-linear state-space models and other non-ideal practical considerations.

7. REFERENCES

[1] D. Estrin, G. Pottie and M. Srivastava, "Intrumenting the world with wireless sensor networks," *Proc. ICASSP*, Salt Lake City, UT, May 2001, pp. 2033-2036.

[2] D. Li, K. D. Wong, Y. H. Hu and A. M. Sayeed, "Detection, classification, and tracking of targets," *IEEE Signal Processing Magazine*, vol. 19, Issue 2, March 2002, pp. 17-29.

[3] J. C. B. Ferreira and J. Waldmann, "Covariance Intersection-Based Sensor Fusion for Sounding Rocket Tracking and Impact Area Prediction," *Control Engineering Practice*, Great-Britain, vol. 15, pp. 389–409, 2007.

[4] J. Waldmann, "In-Flight Alignment in INS-Aiding with Switched Feedforward/Feedback of Error Estimates," *19th International Congress of Mechanical Engineering*, COBEM2007, 2007, Braslia, DF, Brazil.

[5] J. Waldmann, "Feedforward INS Aiding: An Investigation of Maneuvers for In-Flight Alignment," *Controle e Automação*, v. 18, pp. 459–470, 2007.

[6] T. Kailath, A. H. Sayed and B. Hassibi *Linear Estimation* Prentice Hall

[7] C. G. Lopes and A. H. Sayed, "Distributed processing over adaptive networks," *Proc. Adaptive Sensor Array Processing Workshop*, MIT Lincoln Laboratory, MA, June 2006.

[8] C. G. Lopes and A. H. Sayed, "Diffusion least-mean-squares over adaptive networks," *Proc. ICASSP*, Honolulu, Hawaii, vol. 3, pp. 917-920, April 2007.

[9] C. G. Lopes and A. H. Sayed, "Incremental adaptive strategies over distributed networks," *IEEE Transactions on Signal Processing*, vol. 55, no. 8, pp. 4064-4077, August 2007.

[10] C. G. Lopes and A. H. Sayed, "Diffusion strategies over adaptive networks: formulation and performance analysis," *IEEE Transactions on Signal Processing*, vol. 56, no. 7, pp. 4064-4077, August 2008.

[11] D. Spanos, R. Olfati-Saber and R. Murray, "Distributed sensor fusion using dynamic consensus," *Proc. 16th IFAC World Congress*, Prague, Czech Republic, Jul. 2005.