16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

DIFFUSION MECHANISMS FOR FIXED-POINT DISTRIBUTED KALMAN SMOOTHING

Federico S. Cattivelli *Ali H. Sayed*

Department of Electrical Engineering
University of California, Los Angeles, CA 90095
Emails: {fcattiv, sayed}@ee.ucla.edu

ABSTRACT

We consider the problem of fixed-point distributed Kalman smoothing, where a set of nodes are required to estimate the initial condition of a certain process based on their measurements of the evolution of the process. Specifically, we consider linear state-space models where the Kalman smoother gives us the MMSE estimate of the initial state of the system. We propose distributed diffusion solutions where nodes communicate with their neighbors and information is propagated through the network via a diffusion process. Hierarchical cooperation schemes are also described.

1. INTRODUCTION

We consider the problem of distributed fixed-point Kalman smoothing (KS). Given a linear state-space system, every node in the network observes measurements of the evolution of the system, and the objective is to collectively estimate the initial state of the system given observations up to the current time. It is well known that for linear state-space models, the optimal estimate in the mean-square error (MSE) sense is given by the Kalman filtering and smoothing algorithms. Distributed Kalman filtering and smoothing have wide range of applications, including target positioning and tracking [1]. At the end of this paper we present an application where we estimate the initial position of a projectile.

The KS problem can be solved in a centralized manner by transmitting all the measurements from the nodes to a central fusion center, which computes the optimal estimate using the KS algorithm, and then relays back the estimates to all nodes. The disadvantage of this method is that it requires large amount of communications between nodes [2].

Distributed *incremental* estimation algorithms have been developed in the context of adaptive filtering for the LMS and RLS algorithms [3, 4]. These algorithms have the disadvantage of requiring a cyclic path through the network. Distributed *diffusion* alternatives of these algorithms have been proposed in [4, 5, 6]. Diffusion algorithms are more amenable to distributed implementations since nodes communicate in an isotropic manner with their neighbors, and no restrictive topology constraints are imposed. Thus the algorithms are easier to implement and also more robust to node and link failure, at the expense of inferior performance compared to incremental or centralized solutions. More recently, diffusion Kalman filtering has been introduced in [7], which forms the basis for our development of diffusion Kalman smoothing. Distributed Kalman filtering has been proposed also in [8] and [9]. Fixed-lag smoothing was also considered in [9].

This material was based on work supported in part by the National Science Foundation under awards ECS-0725441 and ECS-0601266.

2. THE KALMAN SMOOTHER

2.1 The Kalman filter

Consider a state-space model of the form:

$$\begin{array}{rcl} x_{i+1} &=& F_i x_i + G_i n_i + u_i \\ y_i &=& H_i x_i + v_i \end{array} \tag{1}$$

where $x_i \in \mathbb{C}^M$ and $y_i \in \mathbb{C}^{PN}$ denote the state and measurement vectors of the system, respectively, at time i. The signal u_i is a deterministic input, and the signals n_i and v_i denote state and measurement noises, respectively, and are assumed to be zero-mean and white, with covariance matrices denoted by

$$\mathrm{E}\begin{bmatrix} n_i \\ v_i \end{bmatrix}\begin{bmatrix} n_j \\ v_j \end{bmatrix}^* = \begin{bmatrix} Q_i & 0 \\ 0 & R_i \end{bmatrix}\delta_{ij}$$

where * denotes conjugate transposition. The initial state x_0 is assumed to have zero mean, covariance matrix Π_0, and to be uncorrelated with n_i and v_i, for all i.

Let $\hat{x}_{i|j}$ denote the linear minimum mean-squares error estimate of x_i given observations y_k up to and including time j. The Kalman filter in its time- and measurement-update forms can be computed by starting from $\hat{x}_{0|-1} = 0$ and $P_{0|-1} = \Pi_0$ and iterating the following equations [10, 11]:

$$\begin{array}{lrcl} \text{Measurement-Update:} & & & \\ & R_{e,i} &=& R_i + H_i P_{i|i-1} H_i^* \\ & \hat{x}_{i|i} &=& \hat{x}_{i|i-1} + P_{i|i-1} H_i^* R_{e,i}^{-1}[y_i - H_i \hat{x}_{i|i-1}] \\ & P_{i|i} &=& P_{i|i-1} - P_{i|i-1} H_i^* R_{e,i}^{-1} H_i P_{i|i-1} \\ \text{Time-Update:} & & & \\ & \hat{x}_{i+1|i} &=& F_i \hat{x}_{i|i} + u_i \\ & P_{i+1|i} &=& F_i P_{i|i} F_i^* + G_i Q_i G_i^* \end{array} \tag{2}$$

where $P_{i|j}$ denotes the covariance matrix of the estimation error $\tilde{x}_{i|j} \triangleq x_i - \hat{x}_{i|j}$.

2.2 The fixed-point smoother

We now consider a Kalman smoother, where we wish to estimate the state at some fixed time i_0, given all observations up to time i, where $i > i_0$. Consider the innovations at time j:

$$e_j = y_j - \hat{y}_{j|j-1} = y_j - H_j \hat{x}_{j|j-1} = H_j \tilde{x}_{j|j-1} + v_j$$

and its covariance matrix

$$R_{e,j} = \mathrm{E}[e_j e_j^*] = H_j P_{j|j-1} H_j^* + R_j$$

Then from the orthogonality of the innovations we have [10, p.371]

$$\hat{x}_{i_0|i} = \sum_{j=0}^{i} \mathrm{E}[x_{i_0} e_j^*] R_{e,j}^{-1} e_j \tag{3}$$

where $\mathrm{E}[x_{i_0}e_j^*] = P_{i,j}H_j^*$ and $P_{i,j} = \mathrm{E}[\tilde{x}_{i|i-1}\tilde{x}_{j|j-1}^*]$. Moreover,

$$P_{i_0|i} = P_{i_0|i_0-1} - \sum_{j=i_0}^{i} P_{i_0,j}H_j^*R_{e,j}^{-1}H_jP_{i_0,j}^* \qquad (4)$$

In [10, p.373] it is shown that for the standard state-space model (1) with $u_i = 0$, and for $j \geq i$ we have

$$P_{i,j} = P_{i|i-1}\Phi_p^*(j,i)$$

where

$$\Phi_p(j,i) = \begin{cases} F_{p,j-1}F_{p,j-2}...F_{p,i} & j > i \\ I & j = i \end{cases} \qquad (5)$$

and $F_{p,i} = F_i - K_{p,i}H_i$ and $K_{p,i} = F_iP_{i|i-1}H_i^*R_{e,i}^{-1}$. It can be shown that the same equations hold for the case when $u_i \neq 0$. Applying the matrix inversion lemma, it is straightforward to show that

$$F_{p,i} = F_iP_{i|i}P_{i|i-1}^{-1} \qquad (6)$$

We now look for recursive updates of the quantities $\hat{x}_{i_0|i}$ and $P_{i_0|i}$. We start by defining the matrix

$$M_i \triangleq P_{i_0|i_0-1}\Phi_p^*(i,i_0)P_{i|i-1}^{-1} \qquad (7)$$

From (3) we have for $i \geq i_0$:

$$\begin{aligned} \hat{x}_{i_0|i} &= \hat{x}_{i_0|i-1} + P_{i_0|i_0-1}\Phi_p(i,i_0)^*H_i^*R_{e,i}^{-1}e_i \\ &= \hat{x}_{i_0|i-1} + M_iP_{i|i-1}H_i^*R_{e,i}^{-1}e_i \\ \hat{x}_{i|i} &= \hat{x}_{i|i-1} + P_{i|i-1}H_i^*R_{e,i}^{-1}e_i \end{aligned}$$

The above two equations can be combined to obtain:

$$\hat{x}_{i_0|i} = \hat{x}_{i_0|i-1} + M_i[\hat{x}_{i|i} - \hat{x}_{i|i-1}] \qquad (8)$$

Equation (8) shows that the l.l.m.s.e. estimate of x_{i_0} given all observations up to time i can be obtained recursively given the estimates $\hat{x}_{i|i}$, $\hat{x}_{i|i-1}$ and the covariance matrix of error, $P_{i|i-1} = \mathrm{E}[\tilde{x}_{i|i-1}\tilde{x}_{i|i-1}^*]$.

From (4) we have for $i \geq i_0$:

$$\begin{aligned} P_{i_0|i} &= P_{i_0|i-1} - P_{i_0,i}H_i^*R_{e,i}^{-1}H_iP_{i_0,i}^* \\ &= P_{i_0|i-1} - M_iP_{i|i-1}H_i^*R_{e,i}^{-1}H_iP_{i|i-1}M_i^* \\ P_{i|i} &= P_{i|i-1} - P_{i|i-1}H_i^*R_{e,i}^{-1}H_iP_{i|i-1}^* \end{aligned}$$

The above two equations can be combined to obtain:

$$P_{i_0|i} = P_{i_0|i-1} + M_i(P_{i|i} - P_{i|i-1})M_i^* \qquad (9)$$

We now need to compute a recursion for the matrix M_i. From (5), (6), and (7) we have

$$\begin{aligned} M_{i+1} &= P_{i_0|i_0-1}\Phi_p^*(i+1,i_0)P_{i+1|i}^{-1} \\ &= P_{i_0|i_0-1}^*\Phi_p(i,i_0)F_{p,i}^*P_{i+1|i}^{-1} \qquad (10) \\ &= M_iP_{i|i}F_i^*P_{i+1|i}^{-1} \qquad (11) \end{aligned}$$

Equations (8), (9) and (11), together with the Kalman filter recursions (2), give a set of recursions that allow us to compute the estimate of x_{i_0} given observations up to time $i > i_0$, given the initial estimate $\hat{x}_{i_0}$ and the error covariance matrix P_{i_0}. The initial condition for M_i is $M_{i_0} = I$.

3. DISTRIBUTED FIXED-POINT SMOOTHER

Consider the case where a set of N nodes are spatially distributed over some region. We may represent the nodes and links between nodes as the vertices and edges respectively of a graph $\mathcal{G} = \{\mathcal{V},\mathcal{E}\}$. Throughout our work we assume that such a graph is connected. Let $\mathcal{N}_k$ denote the closed neighborhood of node k (i.e., the set of nodes connected to node k including itself). The *degree* of node k is defined as the number of neighbors of node k including itself. It is assumed that at time i, every node k collects a measurement $y_{k,i} \in \mathbb{C}^{P\times 1}$ according to model (1) as follows:

$$y_{k,i} = H_{k,i}x_i + v_{k,i} \qquad k = 1,...,N \qquad (12)$$

It is assumed that model (1) corresponds to collecting all N measurements from (12) as follows:

$$y_i = \begin{bmatrix} y_{1,i} \\ \vdots \\ y_{N,i} \end{bmatrix} \quad H_i = \begin{bmatrix} H_{1,i} \\ \vdots \\ H_{N,i} \end{bmatrix} \quad v_i = \begin{bmatrix} v_{1,i} \\ \vdots \\ v_{N,i} \end{bmatrix} \qquad (13)$$

We further assume that the measurement noises $v_{k,i}$ are spatially uncorrelated, i.e.,

$$\mathrm{E}\begin{bmatrix} n_i \\ v_{k,i} \end{bmatrix}\begin{bmatrix} n_j \\ v_{l,j} \end{bmatrix}^* = \begin{bmatrix} Q_i & 0 \\ 0 & R_{k,i} \end{bmatrix}\delta_{ij}\delta_{kl}$$

The objective in a distributed smoother implementation is for every node k in the network to compute an estimate of the unknown state x_{i_0}, while sharing data only with its neighbors. We will denote the estimate of x_{i_0} obtained by node k given observations up to time i as $\hat{x}_{k,i_0|i}$. It is also desirable that the quality of this estimate be comparable to the global estimate of $x_{i_0|i}$ had node k had access to all measurements across the entire network and not just its neighborhood.

3.1 Diffusion Kalman smoother

In a diffusion implementation, nodes communicate with their neighbors in an isotropic manner and cooperate to obtain better estimates than they would without cooperation. The diffusion KS algorithm and its variants require the definition of a diffusion matrix $C \in \mathbb{R}^{N\times N}$ with the properties:

$$\mathbb{1}^*C = \mathbb{1}^* \qquad c_{l,k} = 0 \text{ if } l \notin \mathcal{N}_k \qquad c_{l,k} \geq 0,\ \forall l,k \qquad (14)$$

where $\mathbb{1}$ is a $N \times 1$ column vector with unity entries, and $c_{l,k}$ is the l,k element of matrix C. We call C the *diffusion* matrix, since it governs the diffusion process, and plays an important role in the steady-state performance of the network. The entries in C represent the weights that are used by the diffusion algorithm to combine nearby estimates.

The proposed diffusion Kalman smoothing algorithm is based on the diffusion Kalman filtering algorithm proposed in our previous work [7]. This algorithm allows us to recursively compute estimates $\hat{x}_{k,i|i}$ and $\hat{x}_{k,i|i-1}$ for every node k in the network. The index k emphasizes the fact that the estimates of different nodes will be different in general, since they have access to different data. At every time i, each node k exchanges matrices $H_{l,i}$ and $R_{l,i}$ with its neighbors $l \in \mathcal{N}_k$. Then every node runs the so-called *incremental update* as follows. Start with $\psi_{k,i} = \hat{x}_{k,i|i-1}$ and $P_{k,i} = P_{k,i|i-1}$ and for every neighboring node $l \in \mathcal{N}_k$, repeat the following in sequential order (recall (2)):

$$\begin{aligned} R_e &\leftarrow R_{l,i} + H_{l,i}P_{k,i}H_{l,i}^* \\ \psi_{k,i} &\leftarrow \psi_{k,i} + P_{k,i}H_{l,i}^*R_e^{-1}[y_{l,i} - H_{l,i}\psi_{k,i}] \\ P_{k,i} &\leftarrow P_{k,i} - P_{k,i}H_{l,i}^*R_e^{-1}H_{l,i}P_{k,i} \end{aligned}$$

where the arrow "$\leftarrow$" denotes a sequential, or non-concurrent assignment. At the end of the above update, the nodes will be left with $\psi_{k,i}$, which is an estimate of $\hat{x}_{i|i}$. The algorithm requires running one further step, known as the *diffusion update*, where the estimates $\psi_{k,i}$ are combined in a convex form to obtain $\hat{x}_{k,i|i}$ as follows:

$$\hat{x}_{k,i|i} = \sum_{l\in\mathcal{N}_k} c_{l,k}\psi_{l,i}$$

after which they can run the following updates:

$$\begin{aligned}\hat{x}_{k,i+1|i} &= F_i\hat{x}_{k,i|i} + u_i\\ P_{k,i+1|i} &= F_iP_{k,i|i}F_i^* + G_iQ_iG_i^*\end{aligned}$$

The diffusion step is an attempt to achieve the global performance via local node interactions.

From equations (8) and (11) we know that the Kalman smoother update can be computed by using knowledge of the Kalman filtering variables $\hat{x}_{i|i}$ and $\hat{x}_{i|i-1}$. Thus, the diffusion Kalman smoothing algorithm is derived by adding the recursion for $\hat{x}_{k,i_0|i}$ and $M_{k,i}$ as shown below. We also take into account an input u_i from the model (1). The diffusion Kalman smoothing algorithm is presented below.

Algorithm 1: Diffusion Kalman smoother (time- and measurement-update form)

Consider a state-space model as in (1) and a diffusion matrix as in (14). Start with $\hat{x}_{0|-1} = 0$, $P_{k,0|-1} = \Pi_0$ and $M_{k,i_0} = I$ and at every time instant i, compute:

```
Step 1: Incremental Update:
  $\psi_{k,i} \leftarrow \hat{x}_{k,i|i-1}$
  $P_{k,i} \leftarrow P_{k,i|i-1}$
  for every neighboring node $l \in \mathcal{N}_k$, repeat:
    $R_e \leftarrow R_{l,i} + H_{l,i}P_{k,i}H_{l,i}^*$
    $\psi_{k,i} \leftarrow \psi_{k,i} + P_{k,i}H_{l,i}^*R_e^{-1}\left[y_{l,i} - H_{l,i}\psi_{k,i}\right]$
    $P_{k,i} \leftarrow P_{k,i} - P_{k,i}H_{l,i}^*R_e^{-1}H_{l,i}P_{k,i}$
  end
  $\hat{x}_{k,i|i} \leftarrow \psi_{k,i}$
  $P_{k,i|i} \leftarrow P_{k,i}$
Step 2: Diffusion Update:
  $\hat{x}_{k,i|i} \leftarrow \sum_{l\in\mathcal{N}_k} c_{l,k}\psi_{l,i}$
  $\hat{x}_{k,i+1|i} = F_i\hat{x}_{k,i|i} + u_i$
  $P_{k,i+1|i} = F_iP_{k,i|i}F_i^* + G_iQ_iG_i^*$
  if $i \geq i_0$ :
    $M_{k,i} = M_{k,i-1}P_{k,i-1|i-1}F_i^*P_{k,i|i-1}^{-1}$
    $\hat{x}_{k,i_0|i} = \hat{x}_{k,i_0|i-1} + M_{k,i}(\hat{x}_{k,i|i} - \hat{x}_{k,i|i-1})$
```

Algorithm 2: Diffusion Kalman smoother (information form)

Consider a state-space model as in (1) and a diffusion matrix as in (14). Start with $\hat{x}_{0|-1} = 0$, $P_{k,0|-1} = \Pi_0$ and $M_{k,i_0} = I$ and at every time instant i, compute:

```
Step 1: Incremental Update:
  $S_{k,i} = \sum_{l\in\mathcal{N}_k} H_{l,i}^*R_{l,i}^{-1}H_{l,i}$
  $q_{k,i} = \sum_{l\in\mathcal{N}_k} H_{l,i}^*R_{l,i}^{-1}y_{l,i}$
  $P_{k,i|i}^{-1} = P_{k,i|i-1}^{-1} + S_{k,i}$
  $\psi_{k,i} = \hat{x}_{k,i|i-1} + P_{k,i|i}\left[q_{k,i} - S_{k,i}\hat{x}_{k,i|i-1}\right]$
Step 2: Diffusion Update:
  $\hat{x}_{k,i|i} = \sum_{l\in\mathcal{N}_k} c_{l,k}\psi_{l,i}$
  $\hat{x}_{k,i+1|i} = F_i\hat{x}_{k,i|i} + u_i$
  $P_{k,i+1|i} = F_iP_{k,i|i}F_i^* + G_iQ_iG_i^*$
  if $i \geq i_0$ :
    $M_{k,i} = M_{k,i-1}P_{k,i-1|i-1}F_i^*P_{k,i|i-1}^{-1}$
    $\hat{x}_{k,i_0|i} = \hat{x}_{k,i_0|i-1} + M_{k,i}(\hat{x}_{k,i|i} - \hat{x}_{k,i|i-1})$
```

Algorithm 1 uses the diffusion Kalman filter in time- and measurement-update form, whereas Algorithm 2 uses it in Information form. Both algorithms are mathematically equivalent, and therefore have exactly the same performance in terms of estimation error. Which algorithm is more convenient will depend on the specific application. It is important to note that even though the notation $P_{k,i|i}$ and $P_{k,i|i-1}$ has been retained for simplicity, these two matrices do *not* represent the covariance of the estimation error any longer, since the diffusion update is not taken into account in the recursions for these matrices. Exact expressions for the covariances of the estimates are derived in [7].

Algorithm 1 requires that at every instant i, nodes communicate with their neighbors their measurement matrices $H_{k,i}$, the covariance matrices $R_{k,i}$, and the measurements $y_{k,i}$ for the incremental update, and their pre-estimates $\psi_{k,i}$ for the diffusion update. The total communication requirement for every node and for every measurement is $PM + M + P^2/2 + 3P/2$ complex scalars, and it requires one matrix inversion per incremental update. Also note that, as discussed in [7], communication of $R_{k,i}$ may not be necessary if its Cholesky factor is computed, say $R_{k,i} = L_{k,i}L_{k,i}^*$, and $\bar{H}_{k,i} = L_{k,i}^{-1}H_{k,i}$ and $\bar{y}_{k,i} = L_{k,i}^{-1}y_{k,i}$ are transmitted instead of $H_{k,i}$ and $y_{k,i}$. In this scenario, the algorithm requires transmission of $PM + M + P$ complex scalars per node per measurement. For Algorithm 2, the total communication per measurement per node, is $M^2/2 + M/2 + MP$ scalars, and it requires two matrix inversions per incremental update.

3.2 Hierarchical Kalman Smoother

Up to now we have considered non-hierarchical Kalman smoothing, where nodes communicate with their neighbors in an isotropic manner, and every node does the same type of processing. This setup is very robust to node and link failure, since the network keeps working whenever a node fails. However, we may obtain performance improvement if we exploit hierarchy in the network. That is, if we assign different responsibilities to different nodes, we can reduce the number of communications required as well as improve the performance, as discussed next.

Consider the case where we cluster the nodes in the network, and we designate a cluster leader or *clusterhead* to every cluster, with the following conditions:

- Every node in the network is a neighbor to at least one clusterhead
- Every clusterhead has a degree larger than or equal to the degree of each of its neighbors
- Two clusterheads cannot be neighbors

An algorithm to cluster the network in such a way can be found in [12] (see Fig. 1). When clusterheads change or the network topology changes, the network must adapt itself to identify new clusterheads. An algorithm to address these issues was proposed in [13].

We now consider the network formed only by clusterhead nodes, and denote this network by Level-1 (the network with all the nodes is called Level-0). Note that every clusterhead will be at most three hops away from the closest clusterhead. Thus, we consider two clusterheads as being Level-1 neigh-

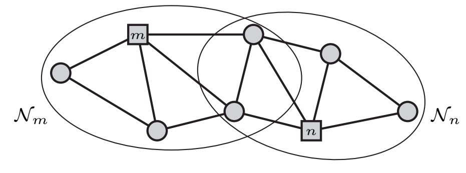

Figure 1: A network where nodes m and n are clusterheads.

bors whenever they are 3 hops or less away from each other. In this setup, communication between the clusterheads may be achieved through multi-hop transmissions.

We now establish a Hierarchical Kalman smoothing algorithm using 2 levels, based on the diffusion Kalman smoother in information form (Algorithm 2). At level 0 (the entire network), nodes exchange data with the clusterheads. These compute the pre-estimates by running the incremental update, and then communicate with their level-1 neighbors to perform a diffusion update. The pseudocode of this algorithm, which we call Algorithm 3, is shown below.

Algorithm 3: Hierarchical Kalman Smoother with 2 levels

```
- Nodes are clustered and clusterheads are designated (e.g. using maximum-degree criterion in [13])
- For every measurement at time $i$, do:
  1. Every node $k$ that is not a clusterhead transmits the quantities $H_{k,i}^*R_{k,i}^{-1}H_{k,i}$ and $H_{k,i}^*R_{k,i}^{-1}y_{k,i}$ to its neighboring clusterheads.
  2. Every clusterhead $k$ updates its estimate $\psi_{k,i}$ and $P_{k,i|i}$ using the incremental update of Algorithm 2.
  3. Neighboring clusterheads in the Level-1 network exchange their estimates $\psi_{k,i}$ and average them as in the diffusion update of Algorithm 2 to obtain $\hat{x}_{k,i|i}$.
  4. Clusterheads compute $\hat{x}_{k,i+1|i}$, $P_{k,i+1|i}$, $M_{k,i}$ and the smoothed estimates $\hat{x}_{k,i_0|i}$ as in the diffusion update of Algorithm 2.
  5. Clusterheads transmit the estimates $\hat{x}_{k,i_0|i}$ to their neighbors. If a node receives more than one estimate, it will select one according to some rule (for instance, the clusterhead with largest degree).
  6. (Optional to improve robustness) Clusterheads transmit $P_{k,i|i}$, $P_{k,i+1|i}$, $M_{k,i}$ and $\hat{x}_{k,i+1|i}$ to one or two neighbors in case of a clusterhead failure.
```

Note that even though the hierarchical method will require some communications to establish the clusterheads and maintain them, it can gain in terms of number of communications. Compared to Algorithms 1 and 2, in Algorithm 3 the nodes only need to communicate with their clusterhead, and not to every other neighbor. In unidirectional communications, this represents savings in terms of energy required for transmission. Also, the multi-hop communications required for the diffusion step do not require much bandwidth since only the estimate is being transmitted.

4. SIMULATIONS

We now apply the diffusion Kalman smoothing algorithms to the problem of estimating the original position of a traveling projectile. This could be useful in applications where a set of sensor nodes are measuring the position of a certain projectile, and they wish to collectively estimate the location of the source of the projectile. In case the source of the projectile is hostile, knowledge of its exact location would aid in evasion or counter-attack strategies.

We assume a simple model of projectile motion, where the acceleration, velocity and position, respectively, are:

$$a = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \quad v = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \quad d = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}$$

and

$$a = \dot{v} \qquad v = \dot{d} \qquad a_x = a_y = 0 \qquad a_z = -g$$

where g is the gravity constant (we use $g = 10$). We formulate a continuous-time state-space model as follows:

$$\underbrace{\begin{bmatrix} \dot{v} \\ \dot{d} \end{bmatrix}}_{\dot{x}} = \underbrace{\begin{bmatrix} 0 & 0 \\ I_3 & 0 \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} v \\ d \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} \begin{bmatrix} 0 \\ 0 \\ -g \end{bmatrix} \\ 0 \end{bmatrix}}_{c}$$

Noting that for the Φ matrix above,

$$e^{\Phi\delta} = I + \delta\Phi \qquad \int_{t_0}^{t_0+\delta} e^{\Phi(t_0+\delta-\tau)d\tau} = \delta I - \delta^2\Phi/2$$

we conclude that the state satisfies the following equation:

$$x(t+\delta) = [I + \delta\Phi]x(t) + [\delta I - \delta^2\Phi/2]c$$

Given a time-step δ, we now define:

$$F \triangleq I + \delta\Phi \qquad \text{and} \qquad u \triangleq [\delta I - \delta^2\Phi/2]c$$

We assume that every node measures the position of the unknown object in either the x and y dimensions, or the x and z dimensions. The assignment of which pair is observable by every node is done at random, but taking care that the three dimensions are observed by at least one node in every neighborhood. Therefore, we have, $H_{k,i} = [0 \quad \text{diag}([1\ 1\ 0])]$ or $H_{k,i} = [0 \quad \text{diag}([1\ 0\ 1])]$.

Denoting $x_i = x(i\delta)$, we arrive at the following discrete state-space model:

$$\begin{aligned} x_{i+1} &= Fx_i + G_i n_i + u \\ y_{k,i} &= H_{k,i}x_i + v_{k,i} \end{aligned}$$

where n_i accounts for modeling errors, and $v_{k,i}$ is the measurement noise at node k.

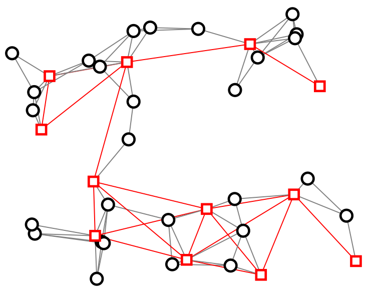

Figure 2: Network topology

In our experiment we use a network with $N = 40$ nodes, a time-step $\delta = 0.1$, $G_i = I$, $Q_i = (0.001)I$, $S_i = 0$ and $R_{k,i} = PR_0$ with $R_0 = 0.5 \times \text{diag}[1 \quad 4 \quad 7]$ and P being a permutation matrix, chosen at random for every node. The diffusion matrix C was chosen such that every neighbor is weighted according to the number of neighbors it has, as follows:

$$c_{lk} = \begin{cases} \alpha_k |\mathcal{N}_l| & \text{if } l \in \mathcal{N}_k \\ 0 & \text{otherwise} \end{cases}$$

where $|\mathcal{N}_k|$ is the cardinality of the closed neighborhood of node k (i.e., the number of neighbors including itself), and α_k is a normalization parameter chosen such that $\mathbb{1}^*C = \mathbb{1}^*$. The results were averaged over 20 independent experiments over the same network topology, which is shown in Fig. 2. The clusterhead nodes in Fig. 2 are represented by squares,

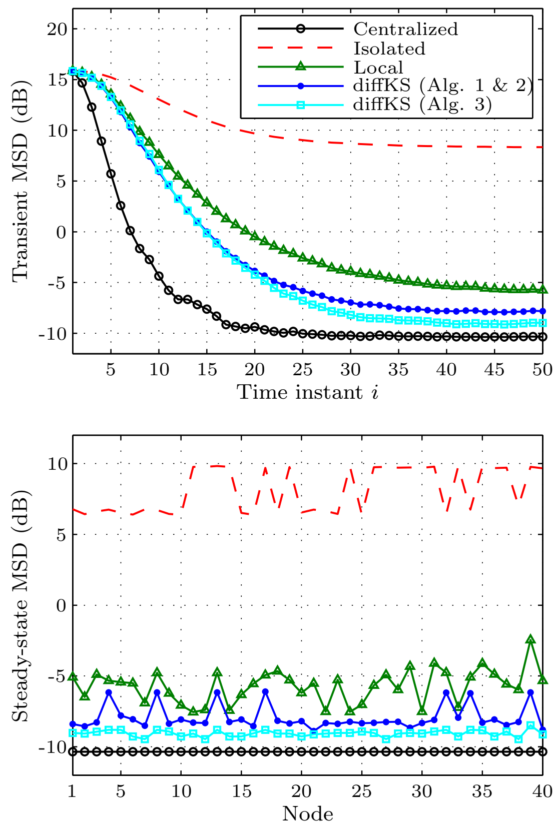

Figure 3: Transient (top) and steady-state (bottom) MSD performance for different algorithms.

and the rest of the nodes by circles. The gray lines represent connections between two nodes. The red lines represent Level-1 (multi-hop) connections between clusterheads.

Figure 3 shows the transient and steady-state mean-square deviation $\mathrm{MSD}_{k,i}$ for five algorithms. In the algorithm denoted "Isolated", all nodes are isolated from the rest of the network, and perform Kalman smoothing using their own measurements. The algorithm denoted "Local" allows communication between neighbors, and every node computes the optimal Kalman smoother given the data from their neighbors only. Also shown is the diffusion Kalman smoother (Alg. 1 and 2), where neighbors not only share their measurements, but also their estimates, and the hierarchical version (Alg. 3). The optimal centralized Kalman smoother algorithm is also shown for comparison. We observe that the diffusion smoothing algorithms have good performance and convergence properties. If we compare the "Local" estimate with Alg. 1 or Alg. 2, we can appreciate the improvement offered by the diffusion step. The improvement of the hierarchical method (Alg. 3) over the non-hierarchical ones is clear at low MSD values. Also note that all the diffusion algorithms have an equalizing effect, whereby all nodes have similar steady-state MSD performance. Finally, Fig. 4 shows the estimates of the trajectory of the object in the z-direction for different algorithms.

5. CONCLUSIONS

We considered fixed-point distributed Kalman smoothing and proposed three diffusion algorithms. Two of them require no hierarchy in the network, whereas the third one clusters the network and assigns cluster leaders that perform most of the processing. The latter algorithm has some performance improvement over the former two, at the expense of higher complexity required to form and maintain the clusters, as well as requiring multi-hop communications. In unidirectional communications, this method also reduces the total amount of communications.

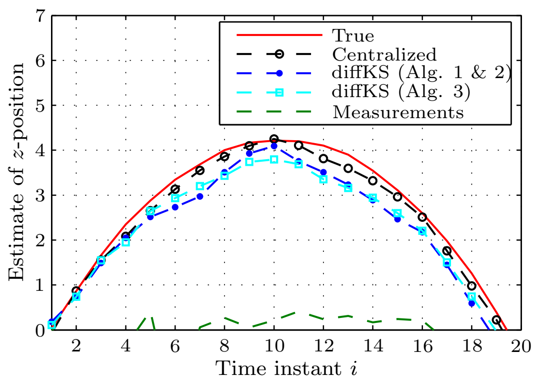

Figure 4: Estimates of vertical position.

REFERENCES

[1] P. Alriksson and A. Rantzer, "Experimental evaluation of a distributed Kalman filter algorithm," in *Proc. 46th IEEE Conf. on Decision and Control*, New Orleans, LA, Dec. 2007.

[2] D. Estrin, L. Girod, G. Pottie, and M. Srivastava, "Instrumenting the world with wireless sensor networks," in *Proc. ICASSP*, Salt Lake City, UT, May 2001, pp. 2033–2036.

[3] A. H. Sayed and C. G. Lopes, "Adaptive processing over distributed networks," *IEICE Trans. on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E90-A, no. 8, pp. 1504–1510, August 2007.

[4] A. H. Sayed and F. Cattivelli, "Distributed adaptive learning mechanisms," in *Handbook on Array Processing and Sensor Networks*, S. Haykin and K. J. Ray Liu, Eds. Wiley, NJ, 2009.

[5] C. G. Lopes and A. H. Sayed, "Diffusion least-mean squares over adaptive networks: Formulation and performance analysis," to appear, *IEEE Trans. on Signal Processing*, 2008.

[6] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion recursive least-squares for distributed estimation over adaptive networks," *IEEE Trans. on Signal Processing*, vol. 56, no. 5, pp. 1865–1877, May 2008.

[7] F. S. Cattivelli, C. G. Lopes, and A. H. Sayed, "Diffusion strategies for distributed Kalman filtering: formulation and performance analysis," in *Proc. Cognitive Information Processing*, Santorini, Greece, June 2008.

[8] R. Olfati-Saber, "Distributed Kalman filtering for sensor networks," in *Proc. 46th IEEE Conf. Decision and Control*, New Orleans, LA, December 2007.

[9] I. D. Schizas, G. B. Giannakis, S. I. Roumeliotis, and A. Ribeiro, "Anytime optimal distributed Kalman filtering and smoothing," in *Proc. IEEE Workshop on Statistical Signal Processing*, Madison, WI, August 2007, pp. 368–372.

[10] T. Kailath, A. H. Sayed, and B. Hassibi, *Linear Estimation*, Prentice Hall, NJ, 2000.

[11] A. H. Sayed, *Fundamentals of Adaptive Filtering*, Wiley, NJ, 2003.

[12] M. Gerla and J. T. C. Tsai, "Multicluster, mobile, multimedia radio network," *Wireless Networks*, vol. 1, no. 3, pp. 255–265, September 1995.

[13] C. Chiang, H. Wu, W. Liu, and M. Gerla, "Routing in clustered multihop, mobile wireless networks," in *Proc. IEEE SICON'97*, April 1997, pp. 197–211.