Distributed Space Situational Awareness (D-SSA) with a Satellite-assisted Collaborative Space Surveillance Network ⋆

Sudharman K. Jayaweera * **R. Scott Erwin** ** **Jed Carty** *

* *Department of Electrical and Computer Engineering, University of New Mexico, Albuquerque, NM, USA (e-mail: jayaweera@ece.unm.edu).*
** *Air Force Research Laboratory, Space Vehicles Directorate, Kirtland AFB, NM 87117.*

Abstract: This paper addresses the problem of tracking space-borne *objects of interest* (OoIs) with a satellite-assisted collaborative space surveillance network (SSN). It is formulated as a *multiple moving-object tracking problem* in a *hybrid distributed wireless sensor network* (WSN) made of both mobile and static nodes. The static nodes may be considered as ground-based sensors with fixed locations, while mobile sensor nodes can be satellites in space. At any given time, due to relative motion among nodes and OoI's, not all sensors are capable of observing a given target. The number of space OoI's (targets) of interest is several orders of magnitude larger than the number of nodes in the SSN, and each node can observe only a finite number of targets at a time. The challenge is to be able to schedule the nodes so that the best possible node-target assignments are made. A distributed approach is more desired in terms of scalability and robustness. However, for a distributed scheduling scheme to be efficient the distributed scheduling decisions need to be consistent across the network, which in turn requires the distributed local tracking estimates to be consistent across the network. As a result, the problem turns out to be a *nonlinear consensus tracking problem on a time-varying graph with incomplete data and noisy communications links.* We propose a distributed and collaborative framework in which each node in the network updates its tracking estimate of an object based on its own observations via an extended Kalman filter (EKF), followed by a consensus update stage in which nodes exchange their local estimates in order to arrive at a consensus on the distributed estimates leading to conflict-free node scheduling. Simulations of a two-dimensional hybrid SSN have been carried out in order to evaluate the performance of the proposed distributed tracking with consensus algorithm. The performance results show that the proposed algorithm indeed performs very well under conditions that can be expected in a realistic SSN, paving way for possible distributed scheduling.

Keywords: Consensus estimation, Distributed tracking, space situational awareness (SSA), space surveillance network (SSN).

1. INTRODUCTION

The problem of tracking space-born *objects of interest* (OoIs) is a challenging problem due to the physical constraints on the available *space surveillance network* (SSN) architecture. For example, the current space surveillance network presumes a handful of ground-based sensor nodes connected in a *centralized architecture*. The number of possible space OoIs that could be present in the space *region of interest* (RoI), on the other hand, may be several magnitudes larger than the number of sensor nodes. There is a great possibility that these fixed sensor nodes may not be able to fully cover the space RoI, leaving some space objects never to be detected. Moreover, some space OoIs may unnecessarily be covered by several sensor nodes, perhaps wasting the network resources. As a solution to these constraints we propose to integrate a small number of space-born *mobile sensor nodes* (i.e. satellites) to the ground-based fixed SSN. Conceivably, the addition of even a small number of satellite sensor nodes in to the SSN can dramatically improve its coverage. In this paper we consider such a satellite-assisted space surveillance network (SA-SSN).

A routine task of an SSN is to assign detected targets to the sensor nodes for the purpose of tracking. This node-target scheduling is important due to the sensing limitations of nodes and the finite maximum number of targets each node can sense/track at any given time. Traditionally, this tracking and node-target scheduling is implemented in a centralized processing architecture. While centralized SSN architecture has its own advantages, a *distributed architecture* may possibly be flexible and more efficient.

⋆ This research was performed while first and third authors were ASEE Air Force Summer Fellows at the Space Vehicles Directorate of the Air Force Research Labs (AFRL) at the Kitland Air Force Base in Albuquerque, NM.

The distributed architecture is easily scalable and perhaps more robust against node failures. However, since not all nodes may be able to observe a given target with the same quality, in a distributed processing architecture different sensor nodes may arrive at different local estimators regarding the *same* space object of interest. This, of course, would then lead to inconsistent distributed node-target scheduling decisions across the nodes in the SSN.

In this paper, we develop a solution to this problem based on distributed but collaborative processing in which sensor nodes are allowed to exchange their local estimators with their neighbors. Specifically, we have developed a *distributed and collaborative non-linear tracking algorithm* complemented by a *consensus updating algorithm* to reach at a consensus tracking estimate that is consistent across the whole (connected) network. As we will show later, the information (in this case, the local estimators) exchange among nodes may help improve the quality of local estimators. More importantly, sufficient number of repeated exchanges can help diffuse information throughout the whole network allowing all distributed estimators to converge to a single *consensus estimator*. Olfati-Saber and Murray (2004); Kashyap et al. (2007). This is an instant of the so-called *consensus in estimation* problem. Kashyap et al. (2007); Kar and Moura (2008). As a result, the distributed and collaborative processing problem in such a hybrid SSN assisted by satellites turns out to be a multiple moving-object *nonlinear tracking with consensus* problem on a *time-varying graph with incomplete data* (to be defined later) and *noisy communications links*.

As mentioned above, according to the proposed distributed SSN architecture assisted by a hybrid sensor network, the distributed tracking with consensus stage is followed by a *distributed scheduling* algorithm that allocates limited network resources for efficient sensing/tracking of space OoI's. In this case, the network resource we are concerned is primarily the processing capability at each node, although it can easily be extended to take into account other aspects. Thus the goal of the distributed scheduling algorithm is to assign each space object to be tracked to the *best* possible sensor nodes subject to a constraint on the maximum number of objects each node can track at any given time. In this paper, due to space limitation, however we consider only the design of distributed non-linear tracking of space OoI's with consensus. The distributed scheduling problem is to be addressed in a follow-up paper separately.

The remainder of this paper is organized as follows: In Section 2 we describe the orbital dynamics of space OoI's and satellite sensor nodes and develop a hybrid WSN to model the situation. This is followed by details on the assumed sensing and communication models. Based on these we next obtain a state-space formulation for multiple moving-object tracking on a time-varying graph with incomplete data and noisy communication links. Section 3 presents the proposed distributed and collaborative tracking with consensus algorithm for the above autonomous hybrid SSN. In Section 4 we present several simulation examples to validate the performance of the proposed distributed tracking with consensus algorithm for a satellite-assisted SSN. In particular, we compare the performance of our algorithm to that attained by a distributed tracking with centralized decision future architecture. Finally, Section 5 concludes the paper by summarizing our contributions.

2. SATELLITE-ASSISTED SPACE SURVEILLANCE NETWORK MODEL

2.1 Network node and target models

Consider a satellite-assisted SSN made of n_m (finite) number of mobile nodes (i.e. satellites) that are on (possibly) different orbits with *known trajectories* and n_s ground-based fixed nodes. We may index the fixed ground nodes as $i = 1, 2, \cdots, n_s$ and the mobile satellites as $i = n_s + 1, n_s+2, \cdots, n_s+n_m$. Let us denote by $\mathcal{N}_s = \{1, 2, \cdots, n_s\}$ and $\mathcal{N}_m = \{n_s + 1, n_s + 2, \cdots, n_s + n_m\}$, respectively, the sets of indices of fixed and mobile nodes respectively. Let $\mathcal{V} = \mathcal{N}_s \cup \mathcal{N}_m$ and denote the total number of sensor nodes in the network by $n = n_s + n_m$. The location of the i-th (fixed or mobile) node at time t is denoted by $\mathbf{r}_i(t)$. With respect to a reference-frame on the earth, the fixed node locations are time-invariant. Hence, $\mathbf{r}_i(t) = \mathbf{r}_i$ for $i = 1, 2, \cdots, n_s$.

The trajectories of the satellite sensor nodes are assumed to be fixed, deterministic and *common knowledge*. Hence, at any given time, each node (be it fixed or mobile) knows its neighbors within its communication radius $R_{c,i}$ which will depend on the type of air interface used, networking protocols (single-hop vs. multi-hop), satellite footprint and, of course, the transmit power level. We assume that the ground-based fixed nodes are connected together over a backbone network. Hence, the communication radius of a fixed node refers to its communication radius with respect to satellite nodes. The sensing radius $R_{s,i}$ of a node depends on the type of sensing that is being used [1] . The set of communication neighbors of node i at time t is denoted by $\Omega_i(t)$. As mentioned above since fixed nodes are wired together, $\mathcal{N}_s \subseteq \Omega_i(t)$, for $i = 1, \cdots, n_s$ and for all t.

The position $\mathbf{r}_m(t)$ of the m-th target space object-of-interest (OoI) relative to the center of the earth is assumed to satisfy the following equation of motion [2] :

$$\ddot{\mathbf{r}}_m = -\frac{\mu}{||\mathbf{r}_m||^2}\mathbf{r}_m + \mathbf{u}_m \tag{1}$$

where μ is the gravitational constant of the earth and $\mathbf{u}_m$ denotes the process noise due to perturbing forces acting on the target object and $||\mathbf{r}_m(t)|| = \sqrt{x_m^2(t) + y_m^2(t) + z_m^2(t)}$ is the distance from the center of the earth to the m-th space OoI at time time t. We may obtain a state-space characterization of the target dynamics by defining the m-th target state $\mathbf{x}^{(m)}(t)$ as $\mathbf{x}^{(m)}(t) = (x_m(t), y_m(t), z_m(t), \dot{x}_m(t), \dot{y}_m(t), \dot{z}_m(t))^T$, so that target state dynamics are governed by the state equation Teixeira et al. (2008)

$$\dot{\mathbf{x}}^{(m)}(t) = \mathbf{f}\left(\mathbf{x}^{(m)}(t)\right) + \mathbf{u}^{(m)}(t). \tag{2}$$

[1] In this work, we are not concerned with the specific type of sensing being used. The algorithms presented here are mostly independent of the type of sensing and thus are easily amenable to many situations.
[2] In fact, the satellite sensor nodes also follow the same equation motion.

where $\mathbf{f}\left(\mathbf{x}^{(m)}(t)\right) = \left(\dot{x}_m(t), \dot{y}_m(t), \dot{z}_m(t), -\frac{\mu}{||\mathbf{r}_m||^2}x_m(t), -\frac{\mu}{||\mathbf{r}_m||^2}y_m(t), -\frac{\mu}{||\mathbf{r}_m||^2}z_m(t)\right)^T$ and $\mathbf{u}^{(m)}(t) = (0,0,0,u_{m,x}(t), u_{m,y}(t), u_{m,z}(t))^T$. While in practice the SSN lives in a 3-dimensional space, most important aspects can easily be considered in two dimensions. Hence, unless stated otherwise, in this paper we use a two-dimensional (2-D) hybrid wireless sensor network to model the above satellite-assisted space surveillance network. Correspondingly, the state $\mathbf{x}^{(m)}(t)$ of the m-th space OoI would be $\mathbf{x}^{(m)}(t) = (x_m(t), y_m(t), \dot{x}_m(t), \dot{y}_m(t))^T$.

2.2 Sensing and observation models

Each sensing node is assumed to have a *sensing beamwidth* of $\beta_{s,i}$, for $i = 1, \cdots, n$. We may denote by $\theta_{s,i}(t)$ the boresight sensing direction of node i at time t, and a node may maneuver the boresight of its sensing beam in any direction $\theta_{s,i}(t) \in [0, 2\pi)$. However, throughout this paper we will assume that the boresight directions are fixed although this has no bearing on the structure of the proposed algorithm. Let us denote by $\alpha_{i,m}(t) = \theta_{s,i}(t) - \angle(\mathbf{r}_i(t) - \mathbf{r}_m(t))$ the angle between the boresight direction of node i and the line connecting sensor node i and the m-th target at time t (i.e. the direction of the vector $\mathbf{r}_m(t) - \mathbf{r}_i(t)$). Denote by $R_{s,i}$ the sensing radius of node i. Then, at any given time instant t, the i-th node can observe the m-th target if and only if

$$|\mathbf{r}_m(t) - \mathbf{r}_i(t)| \leq R_{s,i} \quad \text{and} \quad |\alpha_{i,m}(t)| \leq \frac{\beta_{s,i}}{2}, \quad (3)$$

for $m \in \{1, \cdots, M\}$ and $i \in \mathcal{V}$.

In the following we assume that sensors make both range and angle measurements. We will denote by $\mathbf{y}_i(t)$ a locally observed signal at node i at time t. Assuming that the m-th target satisfies the sensing conditions (3) with respect to node i, the observation at node i on the m-th space OoI at time t will be denoted by $\mathbf{y}_i^{(m)}(t)$:

$$y_i^{(m)}(t) = \begin{bmatrix} \rho(\mathbf{r}_i(t), \mathbf{r}_m(t)) \\ \psi(\mathbf{r}_i(t), \mathbf{r}_m(t)) \end{bmatrix} + \mathbf{v}_i^{(m)}(t) \quad (4)$$

where $\mathbf{v}_i^{(m)}(t)$ is the measurement noise at node i corresponding to the m-th target, $\rho(\mathbf{r}_i(t), \mathbf{r}_m(t)) = \sqrt{(x_m(t) - x_i(t))^2 + (y_m(t) - y_i(t))^2}$ is the range and $\psi(\mathbf{r}_i(t), \mathbf{r}_m(t)) = \arctan\left(\frac{y_m(t)-y_i(t)}{x_m(t)-x_i(t)}\right) - \arctan\left(\frac{y_i(t)}{x_i(t)}\right)$ is the corresponding angle.

Denote by $\mathcal{S}_m(t) \subseteq \mathcal{V}$ the set of sensor node indices that are observing the m-th space OoI at time t and let $N_m^t = |\mathcal{S}_m(t)|$ be the number of sensor nodes that observes the m-th target at time t.

2.3 Communication and hybrid sensor network models

Let us denote by $\beta_{c,i}$ the communications antenna beamwidth of node i. At any given time, the communications antenna boresight direction is denoted by $\theta_{c,i}(t) \in [0, 2\pi)$. For simplicity of exposition, in this paper we assume omni-directional communications antennas so that $\beta_{c,i} = 2\pi$. Hence, at time t a mobile node $i \in \mathcal{N}_m$ can communicate with any other node $i' \in \mathcal{V}$ if $|\mathbf{r}_i(t) - \mathbf{r}_{i'}(t)| \leq R_{c,i}$. Whether this is also a necessary condition depends on whether the network is capable of multi-hop communications or not. In the following treatment, we will indeed assume that all communications are single-hop and thus the condition $|\mathbf{r}_i(t) - \mathbf{r}_{i'}(t)| \leq R_{c,i}$ is both sufficient and necessary.

The graph $G(t)$ of the SSN is time-varying due to the relative movement of the nodes. However, since, as mentioned above, fixed nodes are connected over a backbone network, node i can always communicate with node i' if $i, i' \in \mathcal{N}_s$. The mobile nodes though may or may not have active links with other nodes (be they mobile or static) at any given time t. With that, the hybrid SSN may be modeled as a time-varying, oriented graph $G(t) = (\mathcal{V}, \mathcal{E}(t))$ where edges $e_{ii'} \in \mathcal{E}(t)$ if node i is connected to node i' at time t.

3. DISTRIBUTED AND COLLABORATIVE TRACKING AND CONSENSUS ALGORITHM FOR AN AUTONOMOUS SSN

In the following, we develop a distributed processing framework for the above satellite-assisted hybrid SSN in which all nodes are autonomous. We propose information sharing among neighboring nodes thus leading to an autonomous architecture in which information processing is both distributed and collaborative.

According to our proposed framework, each node i locally processes its observations $\mathbf{y}_i(t)$ before sharing it with other nodes. This involves making local decisions, or forming local estimates. The information exchange among autonomous nodes has two objectives: First, the information sharing is helpful in terms of improving upon local state estimates. More importantly, however, the information sharing becomes critical when nodes need to make distributed and autonomous decisions on actions that have a global impact. An example, in this context, is when nodes need to make autonomous decisions on maneuvering in order to decide who should track what objects. We term this as *distributed scheduling*, as opposed to *centralized scheduling* in which a central node makes all such global decisions. To avoid waste of network resources and conflicts *distributed scheduling* clearly requires that all local estimators are in agreement with each other.

We assume that nodes take observation samples at $t = kh$, for $k = 0, 1, 2, \cdots$, and generate local estimates at the rate of $\frac{1}{h}$. The information exchange period among nodes is denoted by T_c, where we assume that $T_c \ll h$. Note that, in the context of SSN this is a reasonable assumption. For the simplicity of notation, let us assume that $h = JT_c$ so that in between each data sample nodes may perform $J \gg 1$ information exchanges [3]. We assume that the communication is always one-hop, so that the communication at each time instant $t = kh$ is only allowed

[3] This set-up can easily be generalized to several situations, notably the case in which consensus information sharing is performed only at periodic intervals after several observation samples. In this case, each node i will simply track the object of interest based on its own observation sample sequence $\{\mathbf{y}_i\}$ according to the tracking-update step below, punctuated by consensus information sharing at regular intervals only after many tracking-update steps.

among neighbors. Since all nodes do not have access to the same information, the resulting state estimates at each node can, in general, be different.

At each time instant $t = kh$ the processing at each node is performed in two stages: (a) *tracking-update*, followed by (b) *consensus-update* Jayaweera (2009); Jayaweera et al. (2010). The tracking-update stage is comprised of two steps: prediction and filtering. The state-estimator at node i for the m-th target after tracking- and consensus-update stages are denoted, respectively, by $\hat{\mathbf{x}}_i^{(m)}(t)$ and $\bar{\mathbf{x}}_i^{(m)}(t)$. In the following, we describe the proposed distributed and collaborative processing architecture. For simplicity of notation, we drop the superscript m denoting the target index since the algorithm is applied for each target separately.

3.1 Tracking-update step

Let us assume that at time $t = kh$, node i has its *consensus-updated* state estimate $\bar{\mathbf{x}}_i(k)$ of a target with the associated covariance matrix $\bar{\mathbf{P}}_i(k)$ (obtained at the end of consensus update stage, explained below). The measurement update process is performed in two steps as is customary in sampled-data EKF.

- ***Prediction step at node*** i***:*** For $t \in [kh, (k+1)h]$, node i performs prediction update as follows:

$$\dot{\hat{\mathbf{x}}}_i(t|kh) = \mathbf{f}\left(\hat{\mathbf{x}}_i(t|kh)\right) \quad \text{for} \quad t \in [kh, (k+1)h] \tag{5}$$

$$\dot{\hat{\mathbf{P}}}_i(t|kh) = \mathbf{F}_i(t)\hat{\mathbf{P}}_i(t|kh) + \hat{\mathbf{P}}_i(t|kh)\mathbf{F}_i^T(t) + \mathbf{Q}_u \quad \text{for} \quad t \in [kh, (k+1)h] \tag{6}$$

with initial conditions $\hat{\mathbf{x}}_i(kh|kh) = \bar{\mathbf{x}}_i(k)$ and $\hat{\mathbf{P}}_i(kh|kh) = \bar{\mathbf{P}}_i(k)$, respectively. As usual, $\hat{\mathbf{P}}_i(t|kh)$ is the approximation to the true prediction error covariance matrix defined as $\mathbf{P}_i(t|kh) = \mathbb{E}\left\{\left(\mathbf{x}(t) - \hat{\mathbf{x}}_i(t|kh)\right)\left(\mathbf{x}(t) - \hat{\mathbf{x}}_i(t|kh)\right)^T\right\}$ obtained via *linearization* around the last consensus-updated estimate $\bar{\mathbf{x}}_i(k)$ at node i, and $\mathbf{F}_i(t)$ is the Jacobian matrix of the non-linear transformation $\mathbf{f}(.)$ along the predicted trajectory $\hat{\mathbf{x}}_i(t|kh)$ of (5), given by

$$\mathbf{F}_i(t) = \left.\frac{\partial \mathbf{f}\left(\mathbf{x}(t)\right)}{\partial \mathbf{x}(t)}\right|_{\mathbf{x}(t)=\hat{\mathbf{x}}_i(t|kh)}.$$

- ***Filtering step at node*** $i \in \mathcal{S}_m(t)$***:***
  Note that, filtering step is only carried out at nodes that observe a given target. Nodes that do not have measurements of a target m at time $t = (k+1)h$ will simply skip the filtering step. As before, let $\hat{\mathbf{x}}_i(k+1|k) = \hat{\mathbf{x}}_i(t|kh)|_{t=(k+1)h}$. If node $i \in \mathcal{S}_m(t)$, then the EKF filtering step is performed as shown in (7) - (9) (we have dropped the superscripts m), where $\mathbf{K}_i(k+1)$ is the extended Kalman filter gain at time $t = (k+1)h$ at node i, $[\Sigma_v]_{ii}(k+1)$ is the covariance of the observation noise $\mathbf{v}_i(t)$ at time $t = (k+1)h$ and $\mathbf{H}_i(k+1)$ is the Jacobian matrix of the non-linear transformation corresponding to the observation, evaluated at the predicted state estimate $\hat{\mathbf{x}}_i(k+1|k)$ at time $t = (k+1)h$.

3.2 Consensus-update step

We assume that there are J consensus exchanges in between each sampled data observation [4]. At the end of the tracking-update stage, each node i has either a filtered or predicted state estimate with the associated covariance matrices. During the consensus update stage, nodes repeatedly exchange their local estimates with their one-step neighbors and update the local estimates. However, we note that only the nodes that had an observation of a particular target, and thus holds a filtered estimate of that target, carries any new information in its local estimate. Thus, during the consensus iteration that follows the k-th tracking update stage, only an estimate that has some dependence on the observations at time k is identified as an *admissible local estimate*. In order to reduce the effect of (communication) noise accumulation, only those nodes with admissible local estimates are allowed to share their local estimates withe their neighbors. Let $\mathcal{S}^{k,j}$ denote the set of nodes that has admissible local estimates to be shared with others at the beginning of iteration j, for $j = 1, \cdots, J$, that follows the k-th tracking step, where we have again dropped the index m denoting a particular target. We call $\mathcal{S}^{k,j}$ the *admissible node set*.

As mentioned above, due to time-varying topology of the network graph, some nodes do not observe certain targets at a given time. At $j = 0$, these nodes thus do not have an admissible local estimate $\hat{\mathbf{x}}_i(k+1|k+1)$ to be shared with other nodes. However, as information exchange among nodes progresses some of these nodes may be able to form their own local estimates by combining what they receive from others with their (local) pure predictors, thereby allowing them to join the admissible node set.

For clarity of notation, let us denote the network graph $G(t) = (\mathcal{V}, \mathcal{E}(t))$ at time $t = kh + (j-1)T_c$ by $G(k,j) = (\mathcal{V}, \mathcal{E}(k,j))$. Let $D(k,j)$ be the diagonal degree matrix of the network graph at time $t = kh + (j-1)T_c$. The adjacency matrix of the graph $G(k,j)$ is denoted by $A(k,j) = [A_{ii'}(k,j)]$, where $A_{ii'}(k,j) = 1$, if the edge $(i,i') \in \mathcal{E}(k,j)$, and $A_{ii'}(k,j) = 0$ otherwise. The graph Laplacian matrix is then defined to be $L(k,j) = D(k,j) - A(k,j)$.

It is important to note that, since only the admissible nodes transmit their local estimates to the others, the *effective network graph* for the consensus exchanges is different from the actual network graph $G(k,j)$. Let us consider the j-th consensus iteration that follows the k-th tracking step so that $t = kh + (j-1)T_c$. Let $\mathcal{S}^{k,j}$ be the admissible node set. Then, the effective network graph $\tilde{G}(k,j) = (\mathcal{V}, \tilde{E}(k,j))$ at time $t = kh + (j-1)T_c$ corresponding to the network $G(k,j)$ with the admissible node set $\mathcal{S}^{k,j}$ is obtained by removing all outgoing edges of nodes that are not in $\mathcal{S}^{k,j}$. Then, it can be shown that the adjacency matrix $\mathtt{A}(k,j)$ of the effective network graph $\tilde{G}(k,j)$ is given by $\mathtt{A}(k,j) = A(k,j)\mathbf{I}_{\mathcal{S}^{k,j}}$, where $\mathbf{I}_{\mathcal{S}^{k,j}}$ is an $n \times n$ diagonal matrix generated from the active node set $\mathcal{S}^{k,j}$ as follows:

[4] Of course, for sufficient information diffusion across the whole network, J needs to be large enough. How large J needs to be depends on the algebraic connectivity of the corresponding graph of the network.

$$\mathbf{K}_i(k+1) = \hat{\mathbf{P}}_i(k+1|k)\mathbf{H}_i^T(k+1)\left(\mathbf{H}_i(k+1)\hat{\mathbf{P}}_i(k+1|k)\mathbf{H}_i^T(k+1) + [\Sigma_v]_{ii}(k+1)\right)^{-1} \quad (7)$$

$$\hat{\mathbf{x}}_i(k+1|k+1) = \hat{\mathbf{x}}_i(k+1|k) + \mathbf{K}_i(k+1)\left(y_i(k+1) - d_i\left(\hat{\mathbf{x}}_i(k+1|k)\right)\right) \quad (8)$$

$$\hat{\mathbf{P}}_i(k+1|k+1) = \left(\mathbf{I} - \mathbf{K}_i(k+1)\mathbf{H}_i(k+1)\right)\hat{\mathbf{P}}_i(k+1|k) \quad (9)$$

$$[\mathbf{I}_{\mathcal{S}^{k,j}}]_{ii'} = \begin{cases} 1 \text{ if } i = i' \text{ and } i' \in \mathcal{S}^{k,j} \\ 0 \qquad \text{otherwise} \end{cases} \quad (10)$$

The corresponding degree matrix $\mathtt{D}(k,j)$ can then be obtained by $\mathtt{A}(k,j)$ leading to the Laplacian matrix $\mathtt{L}(k,j) = \mathtt{D}(k,j) - \mathtt{A}(k,j)$.

Let $\bar{\mathbf{x}}_i(k,j)$ be the i-th node's updated local estimate at the j-th consensus iteration that follows the k-th EKF-tracking update step. Let $\bar{\mathbf{X}}(k,j) = [\bar{\mathbf{x}}_1(k,j), \cdots, \bar{\mathbf{x}}_n(k,j)]^T$ be the Nn-length estimator after the j-th consensus update, for $j = 1, \cdots, J$ with the associated covariance matrix $\bar{P}(k,j)$. Note that, $\bar{P}_i(k)$ required in the earlier prediction tracking update step is indeed the i-th $N \times N$ main diagonal block of this $\bar{P}(k,j)$. With these definitions, now we are in position to explain the consensus updating process as below:

First, the consensus iteration loop that follows the k-th tracking update stage is initialized as follows: $\mathcal{S}^{k,0} = \mathcal{S}_m(t)$, for the m-th target. If $i \in \mathcal{S}^{k,0}$, then $\bar{\mathbf{x}}_i(k,0) = \hat{\mathbf{x}}_i(k|k)$. Otherwise, we arbitrarily set $\bar{\mathbf{x}}_i(k,0) = \mathbf{0}$ and $\hat{P}_i(k|k) = \epsilon \mathbf{I}_N$ for some $\epsilon > 0$. Then, the initial covariance matrix $\bar{P}(k,0) = \hat{P}_1(k|k) \oplus \cdots \oplus \hat{P}_n(k|k)$ [5].

Next, note that the received data vector at node i at time $t = kh + (j-1)T_c$ (corresponding to the j-th exchange after the k-th tracking-update), for $j = 1, \cdots, J$, can be written as $\mathbf{z}_i(k,j) = \left(\mathbf{z}_{i,1}^T(k,j), \mathbf{z}_{i,2}^T(k,j), \cdots, \mathbf{z}_{i,i-1}^T(k,j), \mathbf{z}_{i,i}^T(k,j), \mathbf{z}_{i,i+1}^T(k,j), \cdots, \mathbf{z}_{i,n}^T(k,j)\right)^T$ where each $\mathbf{z}_{i,i'}(k,j)$ is a noise corrupted version of the N-vector $\bar{\mathbf{x}}_i(k,j-1)$ except for $i' = i$ in which case there is no noise: i.e. $\mathbf{z}_{i,i'}(k,j) = \bar{\mathbf{x}}_{i'}(k,j-1) + \mathbf{w}_{i',i}(k,j)$, for $j = 0, \cdots, J$, where $\mathbf{w}_{i',i}(k,j)$ denotes the (wide-sense stationary) zero-mean receiver noise at node i in receiving the estimator of node i' and thus $\mathbf{w}_{i,i} = \mathbf{0}$ for all i, where $\mathbf{0}$ denotes an N-vector of all zeros. Let $\mathbb{E}\left\{\mathbf{w}_{i',i}\mathbf{w}_{i',i}^T\right\} = \Sigma_{i',i}$. Note that if either node i' does not have a valid local estimator to be shared or if it is not connected to node i according to the graph topology, then $\mathbf{z}_{i,i'} = \mathbf{0}$.

At the j-th consensus iteration step, each node i forms a linear estimator of the following form as their updated consensus estimate:

$$\bar{\mathbf{x}}_i(k,j) = \bar{\mathbf{x}}_i(k,j-1) + \gamma_i(j)\sum_{i'=1}^{n} \mathtt{A}_{i,i'}(j)\left(\mathbf{z}_{i,i'}(k,j) - \bar{\mathbf{x}}_i(k,j-1)\right) \quad (11)$$

where $\gamma_i(j)$ is a weight coefficient used at node i. While for $\forall i \in \mathcal{S}^j$ these may be chosen arbitrarily (subject to convergence requirements), for $i \in \left(\mathcal{S}^j\right)^c$ and if $\sum_{i'=1}^n \mathtt{A}_{i,i'}(k,j) \neq 0$, we explicitly propose to set $\gamma_i(j) = \left(\sum_{i'=1}^n \mathtt{A}_{i,i'}(k,j)\right)^{-1}$. The reason for this particular choice, as we will see later, is to ensure the unbiasedness of the consensus estimate. Let us define the diagonal matrix $\mathbf{\Gamma}(j) = \mathrm{diag}\left(\gamma_1(j), \cdots, \gamma_n(j)\right)$. With these, it can be shown that the consensus update dynamics are given by the following equation

$$\bar{\mathbf{X}}(k,j) = \bar{\mathbf{X}}(k,j-1) - \left[[\mathbf{\Gamma}(j)\mathtt{L}(k,j)] \otimes I_N\right]\bar{\mathbf{X}}(k,j-1) - \left[\mathbf{\Gamma}(j) \otimes I_N\right]\mathbf{W}(j), \quad (12)$$

where $\otimes$ denotes the Kronecker product, $\mathbf{w}(j) = \left(\mathbf{w}_1(j)^T, \cdots, \mathbf{w}_n(j)^T\right)^T$ with $\mathbf{w}_i(j) = -\sum_{i'=1}^n \mathtt{A}_{i,i'}(k,j)\mathbf{w}i', i(j)$. Recall that $\mathbf{w}_{i',i}$ denotes the receiver noise at node i in receiving the estimator of node i'.

Let us define $\bar{\mathbf{e}}(k,j)$ to be the error vector after the j-th consensus update that followed the k-th tracking step:

$$\bar{\mathbf{e}}(k,j) \triangleq \bar{\mathbf{X}}(k,j) - (\mathbf{1} \otimes \mathbf{I}_N)\,\mathbf{x}(k), \quad (13)$$

It should be noted that in (13) we have assumed that the true state of the target space OoI is constant during the consensus updating process. In reality, the state of the space object will evolve during the consensus-updating process. However, if the information exchange rate much faster compared to the target velocities this will cause negligible difference. If on the other hand, that is not the case, the current algorithm can be modified to account for this state evolution from $t = (k+1)h$ to $t = (k+1)h + (j-1)T_c$ when the j-th update is performed. However, this needs enough care to ensure that a reasonable convergence of the consensus algorithm can be guaranteed, which is beyond the scope of this paper. Note that, the corresponding covariance matrix associated with the error vector $\bar{\mathbf{e}}$ is given by $\bar{\mathbf{P}}(k,j)$. From (12) and (13), it follows that

$$\bar{\mathbf{e}}(k,j) = \left(\mathbf{A}(k,j) \otimes I_N\right)\bar{\mathbf{e}}(k,j-1) - \left[\mathbf{\Gamma}(j) \otimes I_N\right]\mathbf{W}(j) + \left[[\mathbf{A}(k,j) \otimes I_N] - \mathbf{I}\right]\mathbf{x}(k), \quad (14)$$

where $\mathbf{A}(k,j) = \mathbf{I} - \mathbf{\Gamma}(j)\mathtt{L}(k,j)$.

Assuming that the filtered estimate $\hat{\mathbf{X}}(k+1)$ at the end of the measurement update stage is an unbiased estimate, so that $\bar{\mathbf{X}}(0)$ is unbiased, from (14) the unbiasedness in the consensus estimator $\bar{\mathbf{X}}(j)$ can be maintained if the coefficient matrix $\mathbf{A}(k,j)$ satisfies $\left(\left(\mathbf{A} \otimes I_N\right) - \mathbf{I}\right)(\mathbf{1} \otimes \mathbf{I}_N) = \mathbf{0}$, which is equivalent to requiring $\left(\left(\mathbf{A} - I_n\right)\mathbf{1}\right) \otimes \mathbf{I}_N = \mathbf{0}$. From this, it follows that the unbiasedness in the consensus estimator $\bar{\mathbf{X}}(k,j)$ requires that 0 is an eigenvalue of the effective Laplacian matrix $\mathbf{L}(k,j)$ with the associated eigenvector $\mathbf{1}$. If indeed we assume this to be the case, then from (14) it is easily seen that covariance matrix evolved as given in (15).

[5] The assumption here is that at the beginning of the consensus-update step, the distributed estimators at different nodes are statistically uncorrelated. Of course, as the consensus iteration progresses these estimators will tend to be more and more correlated.

$$\bar{\mathbf{P}}(k,j) = (\mathbf{A}\otimes\mathbf{I}_N)\,\bar{\mathbf{P}}(k,j-1)\,(\mathbf{A}\otimes\mathbf{I}_N)^T + \mathbb{E}\left\{[\mathbf{\Gamma}(j)\otimes\mathbf{I}_N]\,\mathbf{w}(j)\mathbf{w}(j)^T\,[\mathbf{\Gamma}(j)\otimes\mathbf{I}_N]^T\right\} \quad (15)$$

3.3 Discussion of the convergence properties of the algorithm

For the simplicity of the algorithm, it is convenient to let $\gamma_i(j) = \gamma(j)$ for $\forall i \in \mathcal{S}^j$. The choice of $\{\gamma_j\}$ needs to be such that the rapid consensus among all estimators is achieved. On the other hand, we believe that to ensure the asymptotic convergence of the consensus estimator (to the constant vector $\mathbf{x}(k)$), the sequence $\{\gamma_j\}$ may need to satisfy the usual necessary conditions of $\sum_{j=0}^{\infty}\gamma_j = \infty$ and $\sum_{j=0}^{\infty}\gamma_j^2 < \infty$. The second condition requires that the sequence γ_j be decaying fast enough to avoid catastrophic noise accumulation Mosquera et al. (2008). Indeed our simulations verify this conjecture.

It should be noted that the structure of the proposed distributed tracking with consensus algorithm is different from the consensus in tracking formulation assumed in Olfati-Saber (2005). Indeed, in our formulation, for a fixed k, the attempt is to achieve asymptotic convergence in terms of j, whereas in Olfati-Saber (2005) essentially $J = 1$: i.e. there is only a single consensus exchange at each k. Since the objective of the distributed SSN is to have consistent state estimates across all nodes at any given time k, our formulation is indeed the appropriate one in this context. It then needs to be pointed out that for a fixed k, the consensus in j is a problem of *consensus in estimation* where as that in Olfati-Saber (2005) is a problem of *consensus in tracking*. Since for each j, for a fixed k, the effective network graph and its Laplacian is different (due to the time-varying admissible node set $\mathcal{S}^{k,j}$ as a function of j), this then turns out to be a consensus in estimation over noisy, time-varying graph.

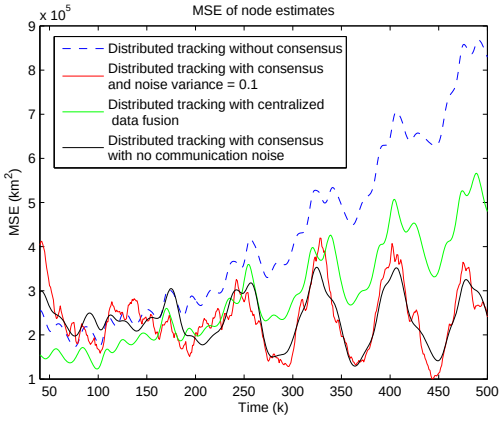


Fig. 1. The MSE performance. $J = 10$ in the consensus algorithm.

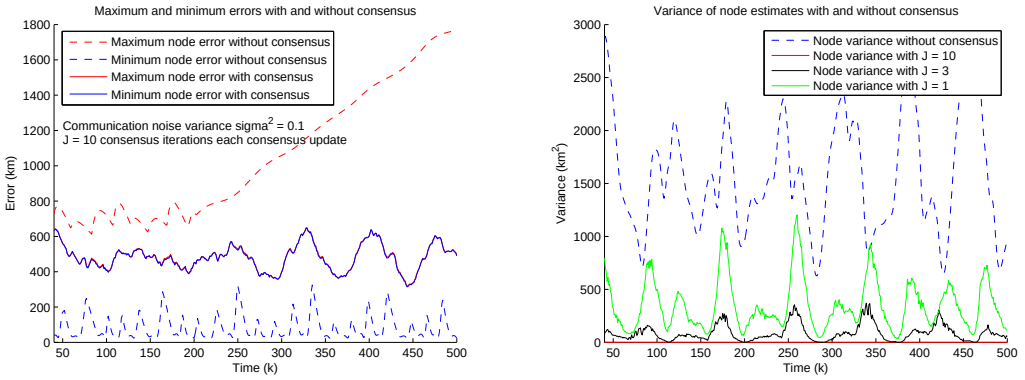
In Jayaweera et al. (2010); Ruan and Jayaweera (2010) it was shown that if the system dynamics and observations were to be linear the above consensus in estimation algorithm (i.e. for a fixed k) is guaranteed to converge to a network-wide consensus in the presence of randomly time-varying, incomplete graph dynamics, as long as each edge is active with non-zero probability. Furthermore, in Ruan and Jayaweera (2010) the authors established analytical conditions on the time-varying graphs under which the asymptotic consensus and steady-state convergence (without communication noise) can be guaranteed. An implication of these results is that if the graph $G(k,j)$ is connected, then each edge of $G(k,j)$ will become active after some point in the sequence of effective network graphs $\left\{\tilde{G}(k,j)\right\}_{j=1}^{\infty}$, thereby guaranteeing the asymptotic convergence of the consensus estimate across our distributed SSN as well.

On the other hand, if there were no communication noise, then a valid question is whether asymptotically in k nodes will also achieve consensus. This was shown to be the case for linear dynamics and observations and $J = 1$ in Olfati-Saber (2005). It can be shown that this still is the case for $J \gg 1$ as long as the system dynamics and observations are linear (details omitted here due to space). In the case of non-linear system dynamics, as in the SSA context as assumed here, it is difficult to provide analytical convergence proofs for the consensus in this sense (i.e. in k). However, in the following section we will use extensive realistic SSA network simulations to demonstrate the effectiveness and convergence of the proposed distributed SSA algorithm.

4. PERFORMANCE OF THE DISTRIBUTED ARCHITECTURE

In all simulations we have assumed 2-D SSN with 5 sensors total, 4 of which are ground based with the other orbiting. The 4 ground based sensors are spaced evenly around the Earth and the orbiting sensor has an orbital radius of 6320 km. For simplicity, we consider a single target of interest that has a slightly eccentric orbit ($e = 0.0597$) with a mean orbital radius of 11776 km. The sample time $T = 172$s for all simulations, and each simulation lasts one sidereal day (86164s).

In Fig. 1 we show the effect of consensus exchanges in bringing the local estimates closer to the actual target location. We use the average Mean-Squared Error (MSE) defined as $\frac{1}{n}\sum_{i=1}^{n}||\hat{\mathbf{x}} - \mathbf{x}||^2$ as a measure of this. Fig. 1 shows this average MSE at each time instant k with the proposed distributed consensus algorithm without any communications noise as well as with noise with $\sigma_c^2 = 0.1$. Also included in Fig. 1 are the average MSE if local nodes run their own local EKF's without any data/estimate exchanges. Moreover, Fig. 1 also shows the MSE if these local estimates were fused by a central fusion node. As seen in 1 indeed the consensus exchange step helps reduce the average MSE of distributed local estimates. Moreover, in some cases, the consensus can help reduce the average MSE even beyond that with central fusion of distributed estimates. This is because with our proposed consensus algorithm thee is the likelihood that at some point even a node which does not have any observations might get a better local estimate due to the information received from other nodes. From then on, this node might be able to predict a reasonably good estimate for the target location. However, without consensus these nodes will always have



(a) (b)

Fig. 2. The impact of information sharing among nodes (a) Maximum and minimum errors with and without consensus. (b) Comparison of the variance of node estimates with and without consensus.

bad estimates and thus fusing them at a central node may not help.

It should be noted that, the objective of the consensus step is to actually bring the distributed node estimates closer to each other, than closer to the actual target state! In Fig. 2a we show the maximum and minimum deviations of the local estimators, across all nodes in the network, from the actual target state if only distributed local EKF algorithms were allowed. It also shows the maximum and minimum deviations from the true target location when the distributed tracking with consensus is performed. First of all note that when the communication noise is sufficiently small, after the consensus the maximum and minimum deviations are almost the same since nodes achieve close-enough consensus. More importantly, as can be seen, in many cases, with our proposed distributed tracking with consensus algorithm the deviation from the actual target state is considerably reduced compared to the worst case deviation some nodes had without consensus exchanges.

As a final performance result, in Fig. 2b we compare the variance of node estimates (across the nodes in the network) with only local EKF estimates and with the proposed algorithm. Note that the variance of node estimates are computed as $\frac{1}{n}\sum_{i=1}^{n}||\hat{\mathbf{x}}||^2 - \left(\frac{1}{n}\sum_{i=1}^{n}\hat{\mathbf{x}}\right)||^2$. As can be seen the disparity among node estimates is vastly reduced due to consensus exchanges. Of course, the effectiveness of the consensus updating depends on the number of consensus iterations J. As can be seen from fig. Fig. 2b even with a single round of consensus exchange, the overall variance among nodes greatly reduced. As the number of iterations increases the variance also decreases ultimately achieving almost perfect consensus (for example, with $J = 10$).

5. CONCLUSION AND FURTHER WORK

A distributed and collaborative tracking algorithm for a space surveillance network embedded with satellite-based sensor nodes was proposed. We formulated the problem as a multiple-object consensus tracking problem on a time-varying graph with incomplete data and noisy communications links. The performance of the proposed distributed processing architecture was evaluated and compared against the achievable performance with a centralized architecture via simulations. As confirmed via our simulation results, the proposed tracking with consensus algorithm significantly reduces the variance of the distributed node estimates. Thus, our proposed framework is a suitable candidate for a system in which scheduling and other decisions are to be made distributively, because local decisions will be based on the (almost) same target estimates that will lead to consistent distributed decisions.

REFERENCES

Jayaweera, S.K. (2009). Distributed space-object tracking and scheduling with a satellite-assisted collaborative space surveillance network. Final project report, AFRL Summer Faculty Fellowship Program.

Jayaweera, S.K., Ruan, Y., and Erwin, R.S. (2010). Distributed tracking with consensus on noisy time-varying graphs with incomplete data. In *10th International Conference on Signal Processing (ICSP'10)*. Beijing, China. Accepted for Publication.

Kar, S. and Moura, J.M.F. (2008). Sensor networks with random links: Topology design for distributed consensus. *IEEE Trans. Signal Process.*, 56(7), 3315–3326.

Kashyap, A., Basar, T., and Srikant, R. (2007). Quantized consensus. *Automatica*, 43(7), 1192–1203.

Mosquera, C., Lopez-Valcarce, R., and Jayaweera, S.K. (2008). Distributed estimation with noisy exchanges. In *IEEE 9th Workshop on Signal Processing Advances in Wireless Communications*. Recife, Brazil.

Olfati-Saber, R. (2005). Distributed kalman filter with embedded consensus filters. In *Proc. 44th IEEE Conf. on Decision and Control*, 8179–8184. Seville, Spain.

Olfati-Saber, R. and Murray, R.M. (2004). Consensus problems in networks of agents with switching topology and time-delays. *IEEE Trans. Autom. Control*, 49(9), 1520–1533.

Ruan, Y. and Jayaweera, S.K. (2010). Performance analysis of distributed tracking with consensus on noisy time-varying graphs. In *5th Advanced Satellite Multimedia System Conference and the 11th Signal Processing for Space Communications Workshop*. Sardinia, Italy. Accepted for Publication.

Teixeira, B., Santillo, M.A., Erwin, R.S., and Bernstein, D.S. (2008). Spacecraft tracking using sampled-data kalman filters. *IEEE Control Sys. Magazine*, 78–94.