Low Computation and Low Latency Algorithms for Distributed Sensor Network Initialization

M. Borkar, V. Cevher and J.H. McClellan

Abstract

In this paper, we show how an underlying system's state vector distribution can be determined in a distributed heterogeneous sensor network with reduced subspace observability at the individual nodes. The presented algorithm can generate the initial state vector distribution for networks with a variety of sensor types as long as the collective set of measurements from all the sensors provides full state observability. Hence the network, as a whole, can be capable of observing the target state vector even if the individual nodes are not capable of observing it locally. Initialization is accomplished through a novel distributed implementation of the particle filter that involves serial particle proposal and weighting strategies that can be accomplished without sharing raw data between individual nodes. If multiple events of interest occur, their individual states can be initialized simultaneously without requiring explicit data association across nodes. The resulting distributions can be used to initialize a variety of distributed joint tracking algorithms. We present two variants of our initialization algorithm: a low complexity implementation and a low latency implementation. To demonstrate the effectiveness of our algorithms we provide simulation results for initializing the states of multiple maneuvering targets in smart sensor networks consisting of acoustic and radar sensors.

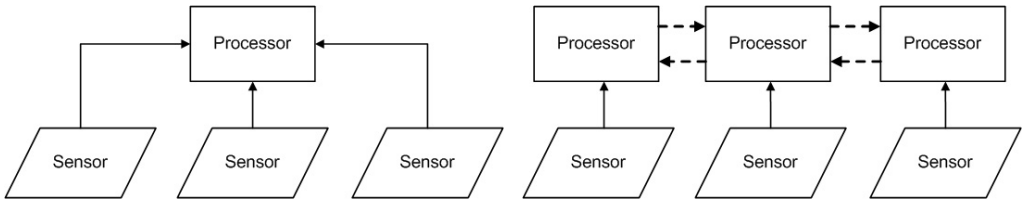
Keywords: Monte Carlo methods, initialization, distributed processing, sensor networks, heterogeneous sensors, data fusion.

M. Borkar and J. H. McClellan are with the Center for Signal and Image Processing, School of ECE, Georgia Institute of Technology, Atlanta GA 30332-0250. V. Cevher is with the Center for Automation Research, University of Maryland, College Park, MD 20742

Prepared through collaborative participation in the Advanced Sensors Consortium sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-02-0008.

I. Introduction

Sensor networks can be classified into two main types based on the processing scheme employed; (i) centralized sensor networks in which the data recorded at the various sensors is transmitted to a central processing unit that is responsible for combining the incoming information and producing meaningful estimates, and (ii) distributed sensor networks in which multiple processing units exist in the network, each of them processing raw data received from some subset of sensors in the network. The distributed processors produce local estimates and share sufficient statistics between each other to produce global estimates. In the extreme case, each sensor node may have its own dedicated processor. This leads to the definition of *smart sensors*. A smart sensor is a node that not only has the ability to sense the environment but also has the ability to process incoming data and communicate with neighboring nodes. In this paper, we will only consider fully decentralized sensor networks consisting of smart sensors. Block diagrams representing centralized and distributed processing are given in Figure 1.



(a) Centralized (b) Distributed

Fig. 1. Centralized vs. distributed processing. The solid lines represent raw data whereas the dashed lines represent sufficient statistics.

In sensor networks, distributed processing is becoming more popular than centralized approaches [1]. This is because centralized networks with only one processing node lose their functionality if the central node is incapacitated. The communication overhead is also significant because when all the sensing nodes try to transmit raw data to the central processing node, the required bandwidth increases significantly with the number of nodes. Sometimes, hybrid approaches are used in which some processing occurs in a distributed manner and the rest occurs in a centralized manner. For example in [2], to reduce the communication load, data at the various nodes is quantized before it is transmitted to a central fusion center. Such a hybrid method is effective in reducing the number of bits transmitted between nodes. However, the bulk of the computational load lies with the central processor and the network can operate only as long as this node survives. To overcome these drawbacks, a distributed processing approach without any central points of failure is attractive.

Based on the types of sensors present, sensor networks can be classified into two categories; (i) homogeneous sensor networks in which all sensor nodes are identical to one another and (ii) heterogeneous sensor networks in which arbitrary sensor nodes observing different modalities of the system are present. Processing data in heterogeneous sensor networks is significantly more complicated than in the homogeneous case since measurements

from different modalities have to be fused together effectively to come up with reliable estimates, a difficult problem if each sensor node observes a different subspace of the system being monitored. However, heterogeneous sensor networks have advantages over their homogeneous counterparts [3]. For example, sensor nodes that can individually observe the state space of the underlying system, referred to as the target state space, may not exist. However, nodes operating in different modalities may collectively be able to observe the target state space. For surveillance applications, heterogeneous networks are also more robust than homogeneous networks since a target could disguise itself and hide from a single sensor modality. As the number of modalities in the network increases, it progressively becomes more difficult for a target to avoid detection by all sensor modalities. Thus heterogeneous sensor networks have the ability to overcome the weaknesses of homogeneous sensor networks.

For the sensor nodes in a distributed sensor network to operate together, efficient and accurate initialization algorithms are vital. This is particularly true in tracking applications. Note that tracking provides a means of reducing computation by focusing the current search space for the phenomena of interest, or target states, near their previous values in a recursive structure. However, to get started, tracking requires an initial state distribution or initialization. These initialization algorithms are faced with the task of collecting local knowledge from individual nodes, fusing these individual components to generate global knowledge, and dispersing this global knowledge throughout the network. In [4], this problem is addressed using hidden Markov random field models. The proposed algorithm can accurately initialize the network locally but the final state estimates may not be globally known. Their approach also requires a stepwise ascent approach which requires multiple communication cycles throughout the network for convergence to a potentially local maximum. Therefore this approach fails if the target state distribution is multimodal, which is the case when multiple targets are present.

The initialization problem can be optimally addressed in a Bayesian framework using belief propagation (BP) methods [5]. In BP methods, the estimated target posterior distribution can be communicated throughout the network and it evolves as the various sensor nodes provide input to the algorithm. Using BP methods in the tracking problem, electrical engineers, computer scientists, and statisticians handle the distributed communication issues by only passing local messages to neighboring nodes and casting the tracking problem in a Markov random field (MRF) framework [6]. Since analytical evaluation of complicated integrals in the BP equations may not be feasible for non-Gaussian potential functions, two almost identical nonparametric methods were independently developed as solutions to this problem. One method is called nonparametric belief propagation (NBP) [7] and the other is called particle message passing (PAMPAS) [8]. In these methods, the messages propagated between nodes are represented by M component mixtures, each component usually being approximated by a Gaussian with a diagonal covariance matrix. Messages received from neighboring nodes and the local knowledge at the current node are combined by multiplying them together. Although both methods use particle sets to represent messages, they primarily differ in

the way these particles are sampled, and this leads to differences in performance. Whereas PAMPAS is specialized for graphical models in which the potentials can be expressed as a mixture of Gaussians, NBP allows more general potential functions. However, the variance estimates generated by NBP tend to be biased upwards of their true values, while those generated by PAMPAS are unbiased.

When combining the received messages with one another and with local information at a node, the direct multiplication of d mixtures consisting of M components each results in a mixture consisting of M^d components. Various methods have been proposed to reduce the computational load at the message product stage. In [7], the Gibbs sampler is used to select a representative sample from the product mixture in order to stop the total number of components from growing exponentially. Use of the Gibbs sampler reduces the computational load to $O(kdM^2)$ operations, where k represents the number of iterations required by the Gibbs sampler to generate one sample. In [9], mixture importance sampling is proposed as a method to reduce computation to $O(dM^2)$ operations. [9] also proposes the use of K-D trees, the computational cost of which is dependent on the choice of various approximating parameters. However, the required computational power is not the only issue with these nonparametric approaches. If the goal is to determine a global state vector distribution, then depending on the network structure, the quality of the final estimate and the rate of convergence could depend on node scheduling, which is the order in which nodes provide input to the algorithm [10]. Another drawback is that neither NBP nor PAMPAS are very robust to missed detections at a subset of nodes in the network. This will be demonstrated through simulation examples.

In this paper, a computationally efficient and robust method for the distributed initializing of the hidden state vector distribution in heterogeneous smart sensor networks is proposed. The desired distribution is referred to as the target state vector distribution. Our algorithm addresses issues related to data fusion and observability. A target's state is initialized by making a discrete approximation to the target's, possibly multimodal, state distribution. The distribution is represented using discrete realizations of the state vector, called particles, and their associated weights. We use a robust weighting strategy that can accommodate missed detections and false alarms. The output of our initialization algorithm can be used to initialize various distributed joint tracking (DJT) algorithms such as those described in [11], [12].

Our algorithm is designed to satisfy certain fundamental constraints to ensure scalability in a distributed network: (i) raw data is not transmitted between nodes, (ii) the data propagated between nodes is the cumulative state information, (iii) nodes are only required to be aware of their own positions and orientations and they need not be aware of the locations or orientations of other nodes, and (iv) the final estimates are unaffected by node scheduling. For simplicity, we assume a fixed one-hop communication path through the network from the first node to the last since specific communication protocols are beyond the scope of this paper. Our algorithm is robust and, with minor modifications, is capable of handling data collisions that may occur when multiple nodes in the network attempt

to initialize the same target simultaneously. The only requirement is that the network must be globally connected, which means that there exists a communication path through the network between any two sensor nodes. The effectiveness of the algorithm is demonstrated in a surveillance scenario using a sensor network consisting of direction-of-arrival (DOA) nodes (e.g., acoustic arrays with known microphone positions) and range-Doppler nodes (e.g., radar sensors). However, the theory behind the algorithm can be extended to arbitrary sensor types.

We first present a computationally efficient version of our algorithm. We refer to this as the *low complexity algorithm*. This algorithm requires $O(M)$ operations at each node for network initialization. Full initialization requires three communication passes through the network; one forward pass to generate a particle support, one reverse pass to determine particle weights and to disseminate particles throughout the network, and another forward pass to disseminate the weights. This algorithm was first developed in [13] and later extended in [14] to compensate for delays entering the system. The derivation of the basic algorithm is repeated in this paper for completeness. The theoretical novelty in this paper is the development of a modified algorithm that can reduce the communication load as well as decrease latency while still satisfying the scalability constraints stated above. We refer to this as the *low latency algorithm*. This modified algorithm can achieve full initialization with only two communication passes; one forward pass to generate particles and weights simultaneously, and one reverse pass to disseminate the particles and weights throughout the network. However, the low latency algorithm comes with a cost of increased computation ($O(M^2)$) at each node. Note that even with the increased computational cost, the low latency implementation has similar, if not lower computational cost when compared to the BP methods discussed above. At the same time, both of our methods are unaffected by node scheduling and demonstrate greater robustness to missed detections than the BP methods.

The organization of the paper is as follows. Section II gives a brief overview of the overall system design. Section III introduces the organic sensors and preprocessors. Section IV discusses our low complexity Monte Carlo approach for the distributed estimation of the target state distribution. Section V focuses on communication between the nodes in the network and the computations required at each node. Section VI introduces our low latency algorithm for Monte Carlo based distributed initialization. Section VII demonstrates the effectiveness of the proposed algorithms on synthetic data and compares performance with BP methods. Conclusions follow in Section VIII.

II. System Design

We define the organic state space for a node as the subspace of the target state space that is observable at that node. In various applications, the organic state space at a node may be a reduced subspace of the target state space. Hence, a one to many mapping may exist from the organic state space to the target state space. For example, when using a node equipped with a radar sensor to localize a target, the organic state space for that node may consist of the target's range from the sensor and its radial velocity, whereas the target's true position may not be observable.

In heterogeneous sensor networks, the organic state spaces may also be dissimilar at different nodes. It is essential to fuse organic estimates from multiple nodes to come up with reliable estimates in the target state space. Such global estimates can be achieved if the network, as a whole, is capable of observing the target state.

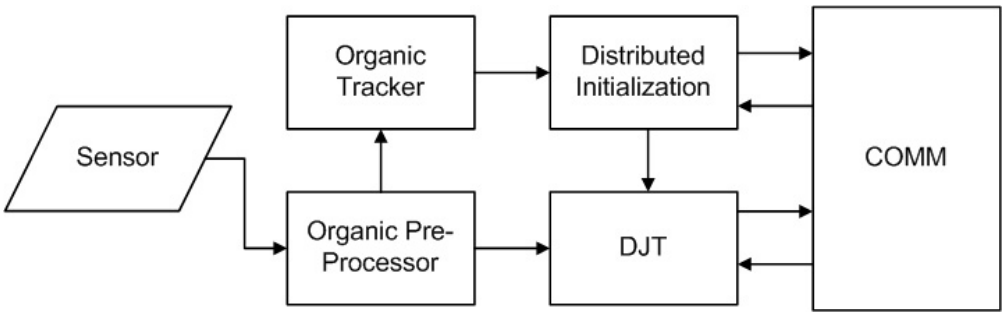


Fig. 2. System block diagram of a smart sensor node.

A block diagram for a smart sensor node is given in Figure 2. The sensor acquires raw data from the environment. This data is fed into the organic pre-processor block which produces state estimates in the organic state space for that sensor node. Depending on the particular sensor modality, the available processing resources and the desired target state space, the organic pre-processor could process the sensed data in various ways to produce organic state estimates for that node. This block could perform beamforming (for sensor arrays), radar pre-processing, and batch processing of measurements to generate motion estimates. The organic state estimates are used to provide input to the DJT block that operates in parallel at the different nodes in the network and tracks the system's varying state parameters in the target state space. An organic tracker also operates within each node, tracking targets in the organic state space for that node using estimates from the organic pre-processor. Though the organic tracker may seem redundant when the DJT is present, it is actually an essential component since it facilitates the detection of new targets at individual nodes. When a target that does not correspond to existing target tracks in the organic tracker is detected, the organic state estimates for that target are fed into the distributed initialization block. The distributed initialization block takes in organic state estimates for new targets from multiple nodes and combines them to produce the initial state estimates in the target state space used by the DJT. This paper will focus on the distributed initialization block. Some of the ideas developed for the initialization algorithm can be extended to the DJT, but such extensions will not be discussed in this document.

III. Organic Sensors and Pre-Processors

We demonstrate our algorithm in a surveillance scenario using a sensor network consisting of DOA nodes and range-Doppler nodes. The goal is to generate probability distributions for multiple targets' states in the $[x\ y\ v_x\ v_y]^T$ space representing the targets' positions and velocities along the x-y directions. We assume that the organic DOA

pre-processors operate in the $[\theta\ Q\ \phi]^T$ space, where θ is the target's bearing, Q is the natural logarithm of the ratio of the target's speed to the target's range, and ϕ is the target's heading direction. In the case of acoustic arrays, which are a type of DOA node, the θ estimate is easily obtained using a beamformer and estimates for Q and ϕ are obtained by processing batches of θ estimates. We also assume that the organic range-Doppler pre-processors operate in the $[r\ v_r]^T$ space where r is the range to the target and v_r is the target's radial velocity. For radar nodes, these estimates can be obtained using radar returns and measuring Doppler shift. Detailed descriptions about these organic pre-processors can be found in [15]–[18]. Even though these particular organic state spaces are used for demonstration purposes, our algorithms are capable of producing accurate initializations even if simpler pre-processors are used. The only condition is that the network, as a whole, should be capable of observing the target state space. Since the range between the target and the DOA node is in general sufficiently larger than the size of the node, the signal from the target can be assumed to be traveling in planar wavefront when it is received at the DOA node. It is also assumed that the radar nodes have hemispherical coverage and are unable to resolve the target's bearing. Note that for the particular sensor nodes considered here, the true location and velocity of any target is not locally observable at any of the individual nodes. Also note that the organic pre-processors operate in dissimilar state spaces having lower dimensionality than the target state space. Hence there is a many to one mapping from the various organic state spaces to the target state space. The sensor network is assumed to be calibrated so that each node is aware of its own location and orientation. However, nodes need not be aware of the locations or orientations of other nodes in the network.

IV. A Low Complexity Monte Carlo Approach for the Distributed Estimation of the Targets' Initial State Distribution

Our initialization algorithm uses a novel Monte Carlo approach to generate an approximation to the target state vector distribution using a weighted set of particles. Following the results in [19], one must sample from the true posterior distribution to generate the optimal particle distribution that minimizes the variance of the weights. Using Bayes' rule, the posterior distribution can be expressed as

$$p(\mathbf{s}_t|\mathbf{z}_t) = \frac{p(\mathbf{z}_t|\mathbf{s}_t)p(\mathbf{s}_t)}{p(\mathbf{z}_t)}, \tag{1}$$

where $\mathbf{s}_t$ represents the target state vector, and $\mathbf{z}_t$ is the vector of organic state estimates from all M nodes at time t. Assuming that the measurements at the individual nodes are independent, conditioned on the current state, the combined data likelihood for all nodes can be factored into the product of the local data likelihoods at the individual nodes. Since the goal is network initialization, prior knowledge of the target state distribution is not available. Hence $p(\mathbf{s}_t)$ is non-informative and is dropped from the equation. Note that $p(\mathbf{z}_t)$ is a proportionality

constant independent of the state. Using this knowledge and the Bayes' rule, (1) can be simplified as

$$p(\mathbf{s}_t|\mathbf{z}_t) \propto \prod_{m=1}^{M} p(\mathbf{z}_{m,t}|\mathbf{s}_t) \propto \prod_{m=1}^{M} p(\mathbf{s}_t|\mathbf{z}_{m,t}), \tag{2}$$

where $\mathbf{z}_{m,t}$ is the set of organic state estimates from the m^{th} node at time t. To limit the communication bandwidth, we choose not to communicate raw data between nodes. Thus, determining the posterior distribution analytically is impossible. Hence we use importance sampling [20] and choose an importance function that can be sampled sequentially at each node without sharing raw data. A logical choice is

$$\pi(\mathbf{s}_t|\mathbf{z}_t) = \frac{1}{M} \sum_{m=1}^{M} p(\mathbf{s}_t|\mathbf{z}_{m,t}), \tag{3}$$

which is an equally weighted mixture of the local posterior distributions at the different nodes. The importance function in (3) has certain features that make it suitable for distributed implementation. First of all, particles can be sampled from this importance function in a distributed manner without sharing raw data between nodes. This feature allows fixed communication bandwidth regardless of the total number of nodes in the network, as will be demonstrated in this section. Another feature of this importance function is that since all the components forming the mixture are equally weighted, the algorithm is unaffected by specific node ordering.

We first show that it is possible to sample particles from the individual posterior distributions at the various nodes in the network. Assume that the target state vector is n-dimensional. Also, assume that at node m, the organic estimates are random realizations of s dimensional feature vectors $\hat{\mathbf{z}}_{m,t}$, where $0 < s \leq n$. If any of these features are not functions of the state vector, they can be discarded since they do not contribute any useful information about the target state vector. Thus, without loss of generality, we can assume that each feature is a function of the state vector.

$$\hat{\mathbf{z}}_{m,t} = f_m(\mathbf{s}_t) = \begin{bmatrix} f_{m,1}(\mathbf{s}_t) \\ f_{m,2}(\mathbf{s}_t) \\ \vdots \\ f_{m,s}(\mathbf{s}_t) \end{bmatrix}. \tag{4}$$

Let $f_m(\cdot)$ be a continuously differentiable vector valued function. If and only if all features in the feature vector at a particular node provide complimentary information without redundancy, then $\nabla f_m(\mathbf{s}_t)$ is nonsingular and

$$\det(\nabla f_m(\mathbf{s}_t)) \neq 0. \tag{5}$$

If $\nabla f_m(\mathbf{s}_t)$ is in fact singular, it would mean that some of the features provide redundant information. In this case, the redundant features can be discarded to give a feature space of reduced dimension and no redundancy. Therefore, without loss of generality, we can assume all features provide complementary information and (5) is satisfied.

Consider the case, when $s < n$. Given $\hat{\mathbf{z}}_{m,t}$, the system in (4) is underdetermined and there exist infinitely many solutions for $\mathbf{s}_t$ satisfying (4). These solutions form a level set in the target state space. In some cases, the level set can be represented by explicit equations relating the state variables. However, in most cases this may not be possible even though the level sets do exist. Let α be any solution of (4). By the implicit function theorem [21], in the neighborhood of α, the level set $L_f(\hat{\mathbf{z}}_{m,t})$ is an $n-s$ dimensional manifold. Let Λ represent the set of all such manifolds. Particles can be generated by sampling uniformly from points in Λ since each of these points is a possible solution to (4).

Now if $s = n$, then by the inverse function theorem [21], given features $\hat{\mathbf{z}}_{m,t}$, a unique inverse function $f_m^{-1}(\cdot)$ exists in the neighborhood of $\hat{\mathbf{z}}_{m,t}$ and therefore there exists a unique solution to (4) given by

$$\mathbf{s}_t = f_m^{-1}(\hat{\mathbf{z}}_{m,t}). \tag{6}$$

Thus the target's true state can be uniquely determined.

Since the organic estimates $\mathbf{z}_{m,t}$ are derived by processing noisy sensor data, we can consider them to be random realizations of the feature vectors. We propose to use the organic estimates at the various nodes as estimates of the true feature vectors in the implementation of the preceding procedure. To account for estimation errors, the proposed particles are randomly perturbed based on the variance determined by the measurement model. In this manner, sets of particles can be sampled locally at each node without sharing raw data.

If node m is a binary detection sensor (i.e., the output is binary based on the probability that a phenomenon of interest occurred) then the mapping from the target state space to the feature space is not differentiable. However, this is a special case of the above since the output is a binary function on some $f_m(\cdot)$. All points in the domain of $f_m(\cdot)$ that result in a detection are possible target states and can be denoted by a set S_d. The same rules for sampling can be applied to points in S_d and the rest of the procedure remains unchanged.

In this paper, we focus on a surveillance scenario. Our network uses DOA nodes and range-Doppler nodes since these sensing modalities represent most commonly used sensor nodes in tracking applications. In this case, particles can be sampled from the individual posterior distributions as given in Table I.

Here $(s_{m,x}, s_{m,y})$ is the m^{th} node's position. Estimates of $(\theta_{m,t}, \sigma_{\theta_{m,t}})$, $(Q_{m,t}, \sigma_{Q_{m,t}})$, and $(\phi_{m,t}, \sigma_{\phi_{m,t}})$ are available from the organic tracker at the m^{th} DOA node. Similarly, estimates of $(r_{m,t}, \sigma_{r_{m,t}})$ and $(v_{r_{m,t}}, \sigma_{v_{r_{m,t}}})$ are available from the organic tracker at the m^{th} range-Doppler node. Since a target's range is not observable at DOA nodes, and a target's bearing is not observable at range-Doppler nodes, these values are drawn from appropriate uniform distributions. Here, $r_{\max}$ is the assumed maximum range at which a target is visible to the DOA node for a given false alarm rate, and $v_{\max}$ is the assumed maximum velocity of a target. Radial velocity is positive if the target is moving away from the node. Tangential velocity is positive in the counterclockwise direction.

	DOA Nodes	**Range-Doppler Nodes**
Step I	$r^{(i)} \sim U\left[0, r_{\max}\right)$	$r^{(i)} \sim N\left(r_{m,t}, \sigma_{r_{m,t}}\right)$
	$\theta^{(i)} \sim N\left(\theta_{m,t}, \sigma_{\theta_{m,t}}\right)$	$\theta^{(i)} \sim U\left[0, 2\pi\right)$
	$Q^{(i)} \sim N\left(Q_{m,t}, \sigma_{Q_{m,t}}\right)$	$v_r^{(i)} \sim N\left(v_{r_{m,t}}, \sigma_{v_{r_{m,t}}}\right)$
	$\phi^{(i)} \sim N\left(\phi_{m,t}, \sigma_{\phi_{m,t}}\right)$	$v_t^{(i)} \sim U\left(-\sqrt{v_{\max}^2 - \left(v_r^{(i)}\right)^2}, \sqrt{v_{\max}^2 - \left(v_r^{(i)}\right)^2}\right)$
Step II	$x_t^{(i)} = r^{(i)} \cos\left(\theta^{(i)}\right) + s_{m,x}$	$x_t^{(i)} = r^{(i)} \cos\left(\theta^{(i)}\right) + s_{m,x}$
	$y_t^{(i)} = r^{(i)} \sin\left(\theta^{(i)}\right) + s_{m,y}$	$y_t^{(i)} = r^{(i)} \sin\left(\theta^{(i)}\right) + s_{m,y}$
	$v_{x_t}^{(i)} = e^{Q^{(i)}} r^{(i)} \cos(\phi^{(i)})$	$v_{x_t}^{(i)} = v_r^{(i)} \cos\left(\theta^{(i)}\right) + v_t^{(i)} \sin\left(\theta^{(i)}\right)$
	$v_{y_t}^{(i)} = e^{Q^{(i)}} r^{(i)} \sin\left(\phi^{(i)}\right)$	$v_{y_t}^{(i)} = v_r^{(i)} \sin\left(\theta^{(i)}\right) - v_t^{(i)} \cos\left(\theta^{(i)}\right)$

TABLE I

SAMPLING PARTICLES IN THE TARGET STATE SPACE FROM LOWER DIMENSIONAL ORGANIC ESTIMATES. STEP I GENERATES PARTICLES IN THE ORGANIC STATE SPACE AND THE UNOBSERVABLE DIMENSIONS ARE SAMPLED UNIFORMLY OVER THEIR DYNAMIC RANGE. STEP II MAPS THE PARTICLES GENERATED IN STEP I TO THE TARGET STATE SPACE.

Following the procedure given in Table I, one could sample particles from the individual posteriors. The combined set of particles must represent (3). Assume that the total number of nodes is M, and D particles are used to generate a discrete approximation to the target state distribution. A simple sampling technique would be to sample D/M particles from each individual posterior and combine these particles to generate the final set of D particles. However, this method has an inherent disadvantage. If one of the nodes does not detect a new target, D/M particles would be spread uniformly over the natural state space for that node. These particles do not provide any useful information to the system. It would be more informative to sample only from the posteriors for the nodes that have detections. Hence, more particles would cover the state space of interest. These disadvantages can be eliminated by using a weighted sampling operation that ensures that the individual posteriors for nodes with detections are equally weighted irrespective of the total number of nodes. This weighted sampling operation does not require synchronization of

the nodes, and it is described in Appendix I.

If these sampled particles are used to make inferences about the posterior distribution, the estimates would be biased due to the discrepancy between the importance function and the true target state distribution. Hence, these particles need to be weighted. Since the data from various nodes is not being shared, the components forming the weights must be computed at each node. To minimize communication, the weights should be transmitted in a cumulative manner. This means that only a fixed number of weights should be transmitted between any pair of sensor nodes and these weights should represent the combined weighting assigned by all preceding nodes in the communication chain.

Following the results of [19], the weights for the particles are given by

$$w_t^{(i)} = \frac{p\left(\mathbf{s}_t^{(i)}|\mathbf{z}_t\right)}{\pi\left(\mathbf{s}_t^{(i)}|\mathbf{z}_t\right)}. \tag{7}$$

Using (2) and (3), (7) can be simplified as follows:

$$w_t^{(i)} \propto \frac{\prod_{m=1}^{M} p\left(\mathbf{s}_t^{(i)}|\mathbf{z}_{m,t}\right)}{\sum_{m=1}^{M} p\left(\mathbf{s}_t^{(i)}|\mathbf{z}_{m,t}\right)} \propto \frac{\prod_{m=1}^{M} p\left(\mathbf{z}_{m,t}|\mathbf{s}_t^{(i)}\right)}{\sum_{m=1}^{M} p\left(\mathbf{s}_t^{(i)}|\mathbf{z}_{m,t}\right)}. \tag{8}$$

Using the Bayes' rule, we obtain

$$p\left(\mathbf{s}_t^{(i)}|\mathbf{z}_{m,t}\right) = \frac{p\left(\mathbf{z}_{m,t}|\mathbf{s}_t^{(i)}\right) p\left(\mathbf{s}_t^{(i)}\right)}{p\left(\mathbf{z}_{m,t}\right)}. \tag{9}$$

Since no prior information about the state vector is available, $p(\mathbf{s}_t)$ is assumed uniform and is dropped from the equation. Thus, (8) simplifies to

$$w_t^{(i)} \propto \frac{\prod_{m=1}^{M} p\left(\mathbf{z}_{m,t}|\mathbf{s}_t^{(i)}\right)}{\sum_{m=1}^{M} \frac{p\left(\mathbf{z}_{m,t}|\mathbf{s}_t^{(i)}\right)}{p(\mathbf{z}_{m,t})}}. \tag{10}$$

Following the given procedure, the particle weights can be calculated, up to a proportionality, by evaluating the ratio of the product of the data likelihoods from the different nodes to a weighted sum of the same likelihoods. Hence, the weights can be updated sequentially if the numerators and denominators are both communicated between nodes. Note that this weighting strategy is unaffected by specific node ordering.

When the final particles are proposed, there is an ambiguity as to which node proposed a particular particle. If a simple Gaussian likelihood function is used and the likelihood for a particle is zero at one of the nodes, then based on (10), its overall weight will also be zero. This situation occurs if even one of the nodes does not detect a target. In such situations, one would not want the overall weight of the particle to be zero since a target may be present with high probability. To avoid this degeneracy, it is important that a robust likelihood function that accounts for missed detections and false alarms is used.

The approach used here is similar to the one used in [22], [23]. Assume that node m generates K organic state estimates. Then, given a particle $\mathbf{s}_t^{(i)}$, the estimates $\mathbf{z}_{m,k,t}$, $k = 1, ..., K$, could have been generated either by a target or by clutter. The clutter distribution is assumed to be Poisson with spatial density λ. The probability of missing the target is set equal to a constant q. It is assumed that there is an equal probability for each of the K organic state estimates to correspond to the true target event, and the organic estimate corresponding to the true target state is Gaussian distributed about that target state. Thus, as shown in [22], the likelihood function can be expressed as

$$p\left(\mathbf{z}_{m,t}|\mathbf{s}_t^{(i)}\right) \propto 1 + \frac{1-q}{\sqrt{(2\pi)^s|\Sigma|}q\lambda K} \cdot \sum_{k=1}^{K} \exp\left\{-0.5\left(\mathbf{z}_{m,k,t} - f_m\left(\mathbf{s}_t^{(i)}\right)\right)^T \mathbf{\Sigma}_{m,t}^{-1}\left(\mathbf{z}_{m,k,t} - f_m\left(\mathbf{s}_t^{(i)}\right)\right)\right\}, \tag{11}$$

where s is the dimensionality of the organic state vector at node m and $\mathbf{\Sigma}_{m,t}$ is the covariance of the Gaussian distribution.

Pseudocode for updating the weights sequentially is given in Appendix I. The final set of particles along with their associated weights give a discrete approximation to the target state vector distribution.

V. Communication and Computation

Using a fixed one hop communication path from the first node to the last node in the network, the low complexity algorithm given in Section IV requires three passes through the communication chain for network initialization.

In the first forward pass, a varying set of a fixed number of D particles representing the equally weighted mixture of posteriors from all preceding nodes is transmitted through the communication chain. Since only those nodes that detect a target are allowed to provide input to the algorithm, a single number n_s representing the accumulating number of nodes with detections is transmitted. At the end of this pass, node M is the only node that has the final set of D particles representing (3).

In the second pass, the communication path is reversed. The final set of D particles are propagated back sequentially to node 1. It was shown that the individual components of the particle weights in (10) can be evaluated independently at each node and the numerator and denominator of the overall weights can be transmitted cumulatively. Thus the data communicated in this pass consists of the set of D particles, the D numerator components, and the D denominator components that form the cumulative weights from the preceding nodes. At the end of the second pass, all nodes in the network have the final set of particles but node 1 is the only node with the final set of weights.

In the third pass, the final set of D weights are dispersed throughout the network using the forward communication path. At the end of this pass, all nodes share the same particles and weights representing the network's global

knowledge.

In a real world implementation, the simple one hop communication protocol can be replaced by more efficient protocols with minor modifications to the proposed algorithms. As long as every node provides its input to the network at the proposal and weighting stages at most one time, the performance of the proposed algorithm will not be affected.

VI. Low Latency Distributed Initialization of a Target's State Distribution

In certain applications, the latency incurred due to three communication passes through the network may not be acceptable. For such applications, we propose a low latency version of the initialization algorithm to enable global initialization with only two communication passes through the network. In the first pass, particles and weights are both sequentially generated. In the second pass, the final particles and weights are disseminated throughout the network. The reduction in communication comes at a cost of an increase in computation at each node; $O(D^2)$ operations at each node for the low latency algorithm compared to $O(D)$ operations at each node for the low complexity algorithm described in Section IV.

Since the first communication pass is used to sequentially generate the particles and weights, the final set of particles and weights available at the output of the m^{th} node, $m = 1, \ldots, M$, must represent the posterior distribution

$$p\left(\mathbf{s}_t|\mathbf{z}_{1,t}, \ldots, \mathbf{z}_{m,t}\right). \tag{12}$$

For the reasons given in Section IV, (3) is still used as an importance function and the particle support is generated sequentially at each node. For this choice of importance function, the particle weights are given by (8). To satisfy these requirements, after processing at node m, the particles must be distributed in accordance with

$$\mathbf{s}_t^{(i)} \sim \frac{1}{m}\sum_{\tilde{m}=1}^{m} p\left(\mathbf{s}_t|\mathbf{z}_{\tilde{m},t}\right), \tag{13}$$

and weights should be assigned according to

$$w_t^{(i)} \propto \frac{\prod_{\tilde{m}=1}^{m} p\left(\mathbf{s}_t^{(i)}|\mathbf{z}_{\tilde{m},t}\right)}{\sum_{\tilde{m}=1}^{m} p\left(\mathbf{s}_t^{(i)}|\mathbf{z}_{\tilde{m},t}\right)}. \tag{14}$$

Using mathematical induction, we show that it is possible to sequentially generate particles and weights according to (13) and (14) respectively to represent the cumulative posterior distribution (12).

Assume that D particles and weights are used to represent the target's state distribution. At node 1, D particles are sampled according to its posterior distribution by following the procedure given in Section IV. These particles are distributed according to

$$\mathbf{s}_t^{(i)} \sim p\left(\mathbf{s}_t|\mathbf{z}_{1,t}\right). \tag{15}$$

For this choice of particle support, the weights are all equal.

$$w_t^{(i)} = \frac{1}{D} \propto \frac{p\left(\mathbf{s}_t^{(i)}|\mathbf{z}_{m,t}\right)}{p\left(\mathbf{s}_t^{(i)}|\mathbf{z}_{m,t}\right)}. \tag{16}$$

It can be clearly seen that (15) and (16) represent (13) and (14) respectively with $m = 1$. Hence these particles and weights together represent the desired posterior distribution (12) with $m = 1$. This set of particles and weights is sent to node 2.

Now consider node m. Node m receives a set of particles $\mathbf{s}_{r,t}^{(i)}$ and weights $w_r^{(i)}$ from node $(m-1)$. Assume that these particles are distributed according to

$$\mathbf{s}_{r,t}^{(i)} \sim \frac{1}{m-1} \sum_{\tilde{m}=1}^{m-1} p\left(\mathbf{s}_t|\mathbf{z}_{\tilde{m},t}\right), \tag{17}$$

and weights are assigned according to

$$w_{r,t}^{(i)} \propto \frac{\prod_{\tilde{m}=1}^{m-1} p\left(\mathbf{s}_{r,t}^{(i)}|\mathbf{z}_{\tilde{m},t}\right)}{\sum_{\tilde{m}=1}^{m-1} p\left(\mathbf{s}_{r,t}^{(i)}|\mathbf{z}_{\tilde{m},t}\right)}. \tag{18}$$

Thus, the received particles and weights together represent the posterior distribution at node $(m-1)$

$$p\left(\mathbf{s}_t|\mathbf{z}_{1,t}, \ldots, \mathbf{z}_{m-1,t}\right). \tag{19}$$

Node m must now provide its own input to the global posterior distribution. It generates a new set of D particles representing its own posterior distribution

$$\mathbf{s}_{n,t}^{(i)} \sim p\left(\mathbf{s}_t|\mathbf{z}_{m,t}\right). \tag{20}$$

The final set of particles and weights after processing at node m must obey (13) and (14). Thus, the new set of particles must be weighted according to

$$w_{n,t}^{(i)} \propto \frac{\prod_{\tilde{m}=1}^{m} p\left(\mathbf{s}_{n,t}^{(i)}|\mathbf{z}_{\tilde{m},t}\right)}{\sum_{\tilde{m}=1}^{m} p\left(\mathbf{s}_{n,t}^{(i)}|\mathbf{z}_{\tilde{m},t}\right)}. \tag{21}$$

Simplifying, we can determine a set of scaled weights $\tilde{w}_{n,t}^{(i)}$ as

$$\tilde{w}_{n,t}^{(i)} = w_{n,t}^{(i)} \cdot \sum_{\tilde{m}=1}^{m} p\left(\mathbf{s}_{n,t}^{(i)}|\mathbf{z}_{\tilde{m},t}\right) \propto p\left(\mathbf{s}_{n,t}^{(i)}|\mathbf{z}_{m,t}\right) \cdot \prod_{\tilde{m}=1}^{m-1} p\left(\mathbf{s}_{n,t}^{(i)}|\mathbf{z}_{\tilde{m},t}\right), \tag{22}$$

where each $\tilde{w}_{n,t}^{(i)}$ needs to be adjusted to account for the particle support. In (22), $p\left(\mathbf{s}_t|\mathbf{z}_{m,t}\right)$ is proportional to the data likelihood at node m, given by (11), and $\prod_{\tilde{m}=1}^{m-1} p\left(\mathbf{s}_{n,t}^{(i)}|\mathbf{z}_{\tilde{m},t}\right)$ can be approximated using the Parzen window

density approximation method [24] as follows

$$\prod_{\tilde{m}=1}^{m-1} p\left(\mathbf{s}_{n,t}^{(i)}|\mathbf{z}_{\tilde{m},t}\right) = \sum_{j=1}^{D} w_{r,t}^{(j)} W\left(\mathbf{s}_{n,t}^{(i)} - \mathbf{s}_{r,t}^{(j)}\right). \tag{23}$$

In (23), $W(\cdot)$ is an appropriate stochastic kernel. In this paper, we choose a Gaussian kernel, the size of which is determined by the particle distribution about the dominant mode.

Using a similar procedure, the particles $\mathbf{s}_{r,t}^{(i)}$ must be weighted according to

$$\hat{w}_{r,t}^{(i)} \propto \frac{\prod_{\tilde{m}=1}^{m} p\left(\mathbf{s}_{r,t}^{(i)}|\mathbf{z}_{\tilde{m},t}\right)}{\sum_{\tilde{m}=1}^{m} p\left(\mathbf{s}_{r,t}^{(i)}|\mathbf{z}_{\tilde{m},t}\right)}. \tag{24}$$

Simplifying, we can determine a set of scaled weights $\tilde{w}_{r,t}^{(i)}$ as

$$\tilde{w}_{r,t}^{(i)} = \hat{w}_{r,t}^{(i)} \cdot \sum_{\tilde{m}=1}^{m} p\left(\mathbf{s}_{r,t}^{(i)}|\mathbf{z}_{\tilde{m},t}\right) \propto p\left(\mathbf{s}_{r,t}^{(i)}|\mathbf{z}_{m,t}\right) \cdot \prod_{\tilde{m}=1}^{m-1} p\left(\mathbf{s}_{r,t}^{(i)}|\mathbf{z}_{\tilde{m},t}\right), \tag{25}$$

where $\prod_{\tilde{m}=1}^{m-1} p\left(\mathbf{s}_{r,t}^{(i)}|\mathbf{z}_{\tilde{m},t}\right)$ can be approximated using the Parzen window density approximation method as follows

$$\prod_{\tilde{m}=1}^{m-1} p\left(\mathbf{s}_{r,t}^{(i)}|\mathbf{z}_{\tilde{m},t}\right) = \sum_{j=1}^{D} w_{r,t}^{(j)} W\left(\mathbf{s}_{r,t}^{(i)} - \mathbf{s}_{r,t}^{(j)}\right). \tag{26}$$

The current set of $2D$ particles $\{\mathbf{s}_{n,t}^{(i)},\ \mathbf{s}_{r,t}^{(i)}\}_{i=1}^{D}$ does not represent (13) since the mixture distribution from which these particles are generated is not equally weighted. This discrepancy can be corrected using the weighted sampling operation given in Appendix II to generate a set of D particles, $\{\mathbf{s}_t^{(i)}\}_{i=1}^{D}$, representing (13).

The scaled weights associated with the particles surviving the sampling operation are stored as $\{\tilde{w}_t^{(i)}\}_{i=1}^{D}$. From (21),(22), (24) and (25), the final set of weights representing (14) can be determined by

$$w_t^{(i)} \propto \frac{\tilde{w}_t^{(i)}}{\sum_{\tilde{m}=1}^{m} p\left(\mathbf{s}_t^{(i)}|\mathbf{z}_{\tilde{m},t}\right)}, \tag{27}$$

where $\sum_{\tilde{m}=1}^{m} p\left(\mathbf{s}_t^{(i)}|\mathbf{z}_{\tilde{m},t}\right)$ can be approximated using the Parzen window density approximation method as follows

$$\sum_{\tilde{m}=1}^{m} p\left(\mathbf{s}_t^{(i)}|\mathbf{z}_{\tilde{m},t}\right) = \frac{1}{D} \sum_{j=1}^{D} W\left(\mathbf{s}_t^{(i)} - \mathbf{s}_t^{(j)}\right). \tag{28}$$

The final set of particles and weights given by $\{\mathbf{s}_t^{(i)},\ w_t^{(i)}\}_{i=1}^{D}$ obeys (13) and (14). Thus, this weighted set of particles represents the posterior distribution (12).

Pseudocode for implementing the low latency algorithm is given in Appendix II.

VII. Simulations

In this section, we demonstrate the effectiveness of our algorithms by generating the initial probability distributions for multiple targets in smart sensor networks consisting of DOA sensors and range-Doppler sensors. We show simulation results for both the low complexity and low latency algorithms described in this paper. Simulations are repeated for a varying number of targets and varying network configurations. BP methods are also simulated and compared with our proposed methods.

Assume that two targets appear simultaneously with initial states given by $\mathbf{s}_1 = [-200,\ -500,\ 10,\ 20]^T$ and $\mathbf{s}_2 = [1600,\ 0,\ -14,\ -14]^T$. The network consists of four nodes; two DOA sensors located at (500 m, 400 m) and (800 m, -300 m), and two range-Doppler sensors located at (200 m, -200 m) and (1200 m, 200 m). The sensor and target positions are displayed in Figure 3.

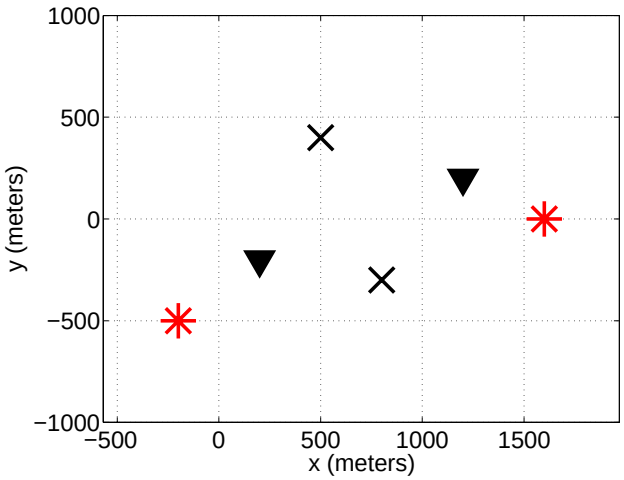


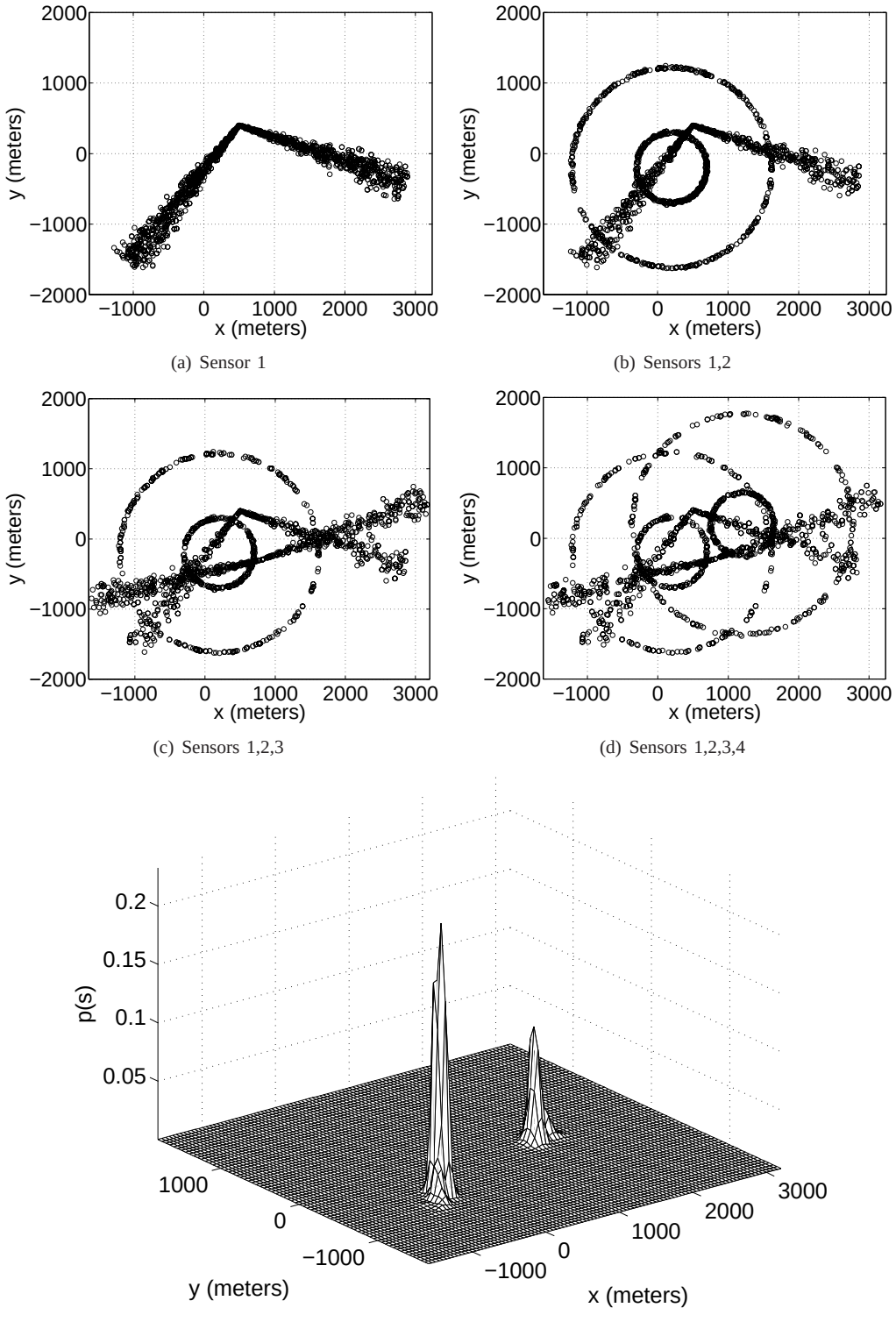
Fig. 3. Sensor and target positions: ▼ represent range-Doppler sensors, $\times$ represent bearing sensors and $*$ represent targets.

In this simulation, $D = 2000$ particles were used to approximate the target state vector distribution. To simulate noisy estimates available from the organic trackers (i.e., $[\theta\ Q\ \phi]^T$ from the DOA trackers and $[r\ v_r]^T$ from the range-Doppler trackers), the estimates provided to each organic tracker are Gaussian distributed about their true values with standard deviations given by

$$\sigma_\theta = 2^\circ,\ \sigma_Q = 0.02\ \text{s}^{-1},\ \sigma_\phi = 8^\circ, \tag{29}$$

$$\sigma_r = 6\ \text{m},\ \sigma_{v_r} = 0.4\ \text{m/s}. \tag{30}$$

The clutter is modelled as a Poisson distributed random variable with parameter $\lambda = 1/7$. The probability of a miss is set equal to 0.1.



(a) Sensor 1

(b) Sensors 1,2

(c) Sensors 1,2,3

(d) Sensors 1,2,3,4

(e) Initial Distribution

Fig. 4. Simulation example for initializing multiple targets using the low complexity algorithm.

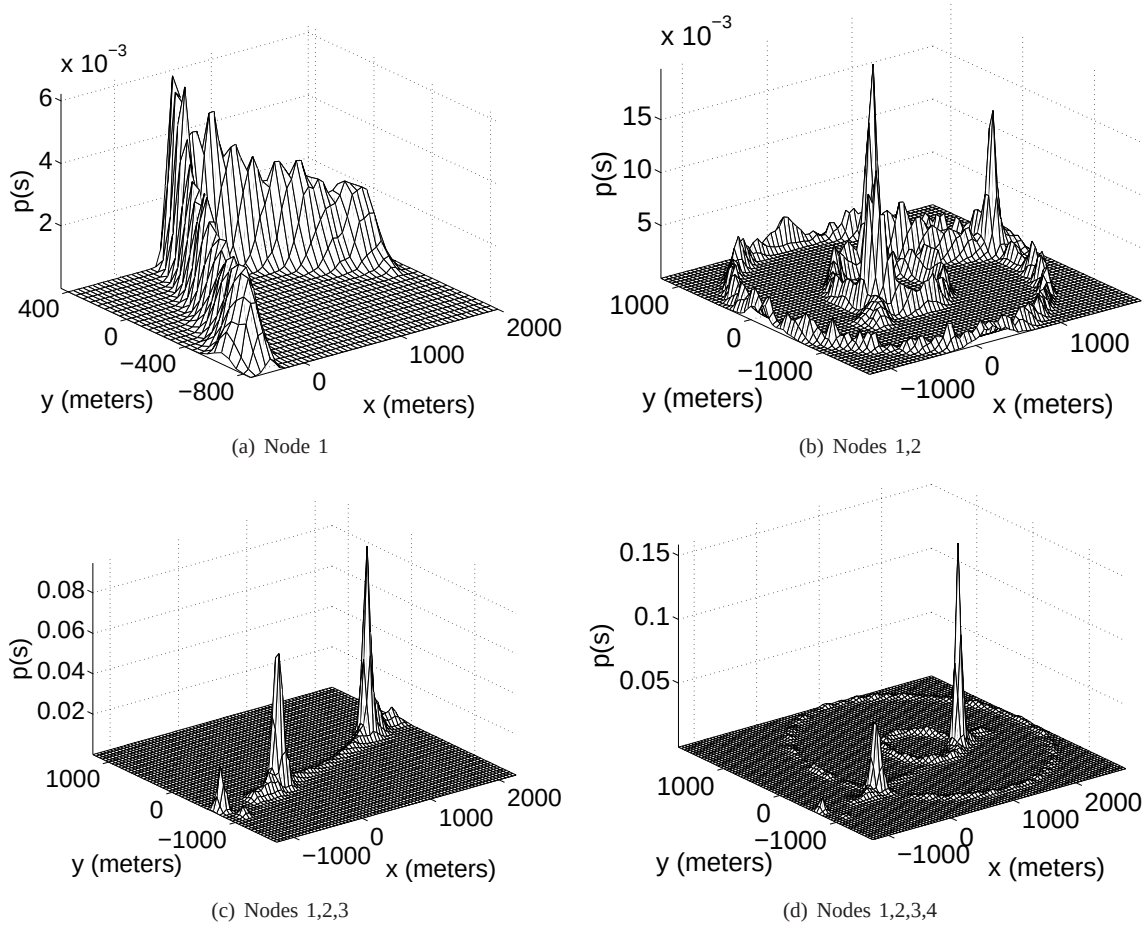
The Gaussianity assumption is made only to simplify the implementation. This is a standard practice for tracking algorithms since the algorithms used to estimate the measurement noise (e.g., Newton recursion, LMS, RANSAC, etc.) assume that the noise is Gaussian. However, it is important to note that even though the assumption of Gaussianity is local at each node, the particle distributions are not necessarily Gaussian in the target state space. This will be seen in the simulations. The presented algorithm can handle non-Gaussian noise sources with minor modifications. The assumption of Gaussian noise affects the initialization algorithm at only two stages. The first impact is at the particle proposal stage. Here, if the noise is non-Gaussian, particles can still be generated by importance sampling. The second impact is at the weighting stage. Here, if the PDF of the noise distribution can be evaluated pointwise, then any arbitrary noise distribution can be used. For simplicity, we will assume that the noise is Gaussian in these simulations.

Figures 4(a) to 4(d) represent the sequential particle proposal stage of the low complexity algorithm. Although the state vector is four dimensional, the subfigures in Figure 4 show only the x-y locations of the particles. In Figure 4(a), node 1, a DOA node, detects the two targets along certain directions and distributes a total of 2000 particles along those directions up to an assumed maximum detection range. These particles are propagated to node 2. Node 2, a range-Doppler node, receives the particles from node 1. These received particles are all assigned a weight of 1 since they represent information from a single node. Node 2 detects the two targets at certain ranges. Since the targets' bearings are not observable, node 2 distributes another 2000 particles about two concentric circles centered at the node position with radii equal to the detected ranges. Out of the 4000 particles, 2000 particles are sampled uniformly with replacement. These particles are shown in Figure 4(b), and are propagated to node 3. Node 3, another DOA node, receives the particles from node 2 and assigns to each of these particles a weight of 2 since they represent the combined knowledge from two nodes. Node 3 distributes another 2000 particles along the directions in which it detects the two targets. These new particles are assigned a weight of 1 since they represent information from node 3 only. From the 4000 particles, a weighted sampling with replacement operation is used to generate 2000 equally weighted particles. These particles are shown in Figure 4(c) and are propagated to node 4. Node 4, another range-Doppler node, receives the particles from node 3 and assigns to each of them a weight of 3 since they represent the combined information from 3 nodes. Then, node 4 detects the targets at certain ranges and distributes another 2000 particles along concentric circles with radii equal to the detection ranges and centered at the node location. These new particles are assigned a weight of 1. From these 4000 particles, 2000 particles are obtained by weighted sampling with replacement. These final particles are plotted in Figure 4(d) and are propagated back to all the nodes.

Weights are calculated for the final particles shown in Figure 4(d). Particles along with their weights are used to generate the PDF of the posterior distribution. The x-y subspace of the posterior distribution is shown in Figure

4(e). As expected, the distribution is highly peaked about the true target states. Estimates of the true target states can be made based on this weighted set of particles. These estimates can be used to initialize any DJT.

The same simulation is repeated using the low latency algorithm. This algorithm propagates the cumulative posterior distribution from node to node by generating particles and weights sequentially at each node. The sequential particle proposal strategy is similar to the one used by the low complexity implementation and is described in detail in the first simulation. Hence, the discussion will not be repeated here.



(a) Node 1

(b) Nodes 1,2

(c) Nodes 1,2,3

(d) Nodes 1,2,3,4

Fig. 5. Simulation example for initializing multiple targets using the low latency algorithm.

Figure 5 demonstrates the low latency algorithm in operation. For ease of view, the PDFs representing the cumulative posterior distributions are generated from the particles and weights and the x-y subspaces of the posterior distributions are plotted. It can be seen that as more nodes provide input to the algorithm, the posterior distribution slowly has its weight concentrated in the areas where the targets are located. The final posterior is shown in Figure 5(d). This distribution is very similar to the posterior obtained using the low complexity algorithm in Figure 4(e). The minor discrepancies are due to the approximations made when the Parzen window density approximation

method is used. However, it is important to note that the dominant peaks remain unchanged regardless of which variation of the algorithm is used.

Next, we simulate NBP and PAMPAS for comparison against our initialization algorithms. The main difference between NBP and PAMPAS is in the message propagation step. This is the step in which the transmitting node generates particles representing its belief about the receiving node's state. In our application, we are interested in determining the global target state vector which is common at every node. Hence no transformation is required from the transmitting node's state to the receiving node's state. This fact eliminates the primary difference between NBP and PAMPAS and facilitates a common implementation, which we shall refer to as the BP implementation. To ensure a fair comparison, the BP implementation was simulated under circumstances that were as close as possible to those under which our initialization algorithms were simulated. We implemented a sequential schedule for inter-node communication as discussed in [10]. In the scenario considered here, this schedule is very similar to the one hop communication protocol used to simulate our algorithms. The presented plots represent the cumulative knowledge available at the last node in the communication chain at the end of the first iteration through the network. Although multiple iterations may be required for the BP implementation to converge, we only simulate a single iteration to make a fair comparison with our algorithm. Following [8], a single outlier particle with zero mean and large covariance was included with each message. This outlier particle was weighted to account for 10% of the entire message probability, which is identical to the probability of miss in the simulation of our initialization algorithm. In practice, Gibbs sampling, mixture importance sampling and K-D trees are used to reduce the computational load of the BP methods during the message product step. To eliminate the effects of these methods on final estimates, we explicitly computed the product messages and performed a weighted sampling on the product mixture to limit the number of transmitted particles.

Figure 6 shows the target state distribution as initialized by the BP implementation. When compared to Figures 4(e) and 5(d), it can be seen that all three distributions have distinct peaks at the true target states. Hence, all three algorithms show similar performance in this simulation.

The simulations in Figure 7 test the various algorithms' robustness to missed detections and specific node scheduling. The same targets and nodes are used, however, in this simulation, the first range-Doppler node only detects the first target and the second range-Doppler node only detects the second target. The DOA nodes detect both targets. The simulations are repeated for an alternate node scheduling which is also sequential but with the communication path reversed.

Figures 7(b) and 7(d) show that in the absence of detections at a subset of the nodes in the network, the BP implementation can fail to accurately initialize the target state distribution in one iteration. In fact, it can also be seen that the final estimate of the target state distribution can depend on node scheduling. In Figure 7(b), the

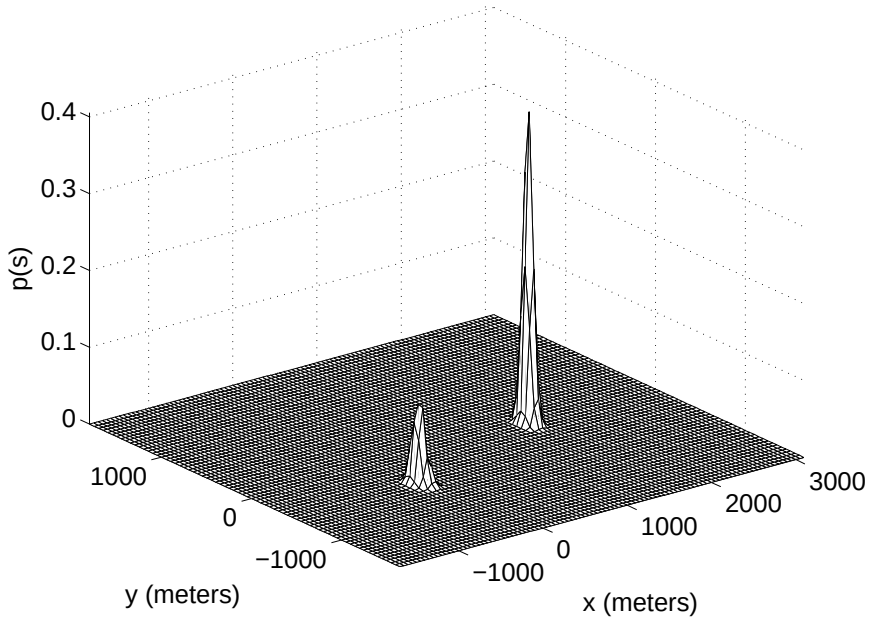
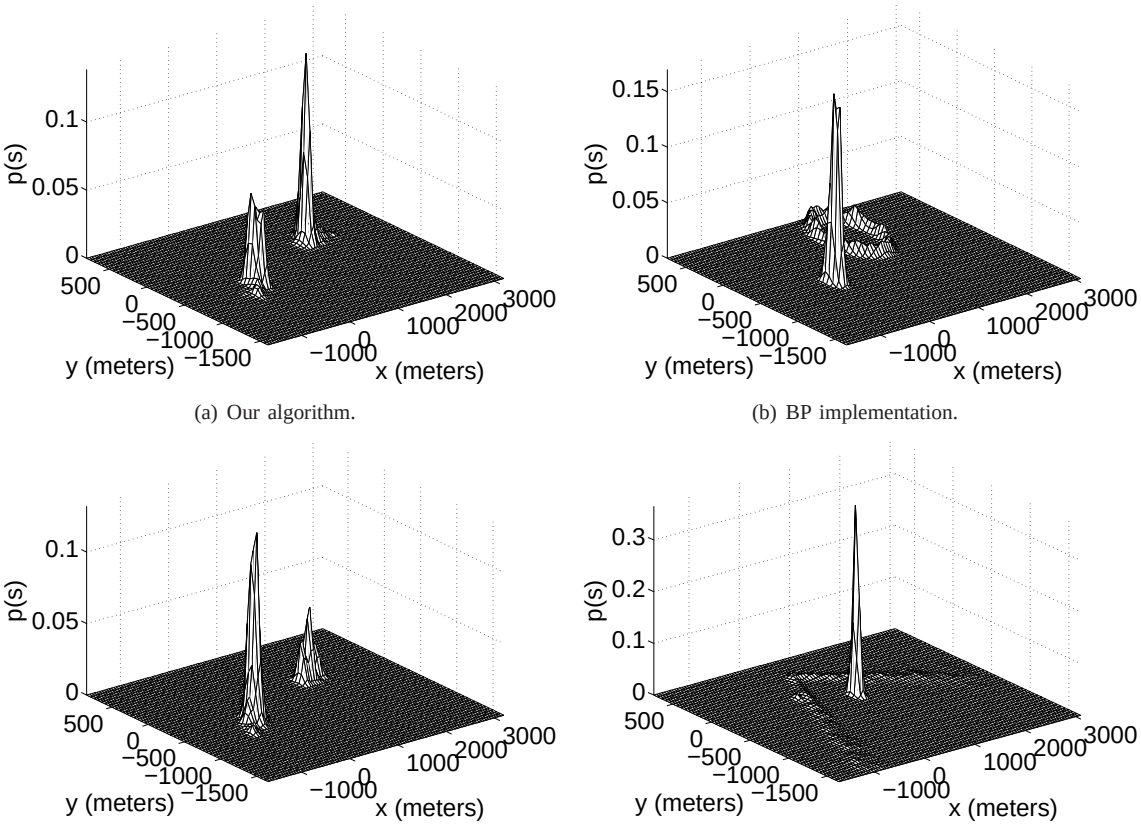


Fig. 6. Simulation example for initializing multiple targets using the BP implementation.

distribution is peaked at the first target's state and there is a large uncertainty about the second target's state. Figure 7(d) shows even worse performance since the one peak that appears in the distribution is not located at either one of the true target states. On the contrary, Figures 7(a) and 7(c) demonstrate the robustness of our algorithm to missed detections and node scheduling. The minor discrepancies are due to different random number realizations. However, two distinct peaks clearly appear at the true target states, indicating the presence of two targets. These plots clearly show the advantages of our initialization algorithm over a comparable BP implementation. Although the BP implementation might converge to the true distribution after a larger number of iterations, we did not simulate additional iterations since the increased expenditure in communication would not justify a fair comparison with our algorithms.

To test the ability of our algorithms to initialize a larger number of targets, both of our algorithms were used to initialize the network with 5 targets present. The node positions remain unchanged and are shown in Figure 8 along with the new target positions. Simulation results are given in Figure 9. For both implementations, the posterior distribution is peaked at the true target states. The low latency algorithm produces some amount of clutter in the distribution. This is due to the approximations made by the Parzen window density approximation method. The final distribution can be further smoothened out by improved kernel selection methods. However, it is clear that both the low complexity and the low latency algorithms are capable of detecting and initializing all 5 targets.

Both of our initialization algorithms were simulated in a larger network with 10 sensor nodes that were positioned as shown in Figure 10. The true target states are given by $\mathbf{s}_1 = [50,\ 250,\ 14,\ 14]^T$ and $\mathbf{s}_2 = [1200,\ -250,\ 14,\ 14]^T$.



(a) Our algorithm.

(b) BP implementation.

(c) Our algorithm. Reverse node ordering.

(d) BP implementation. Reverse node ordering.

Fig. 7. Testing robustness against missed detections and node scheduling.

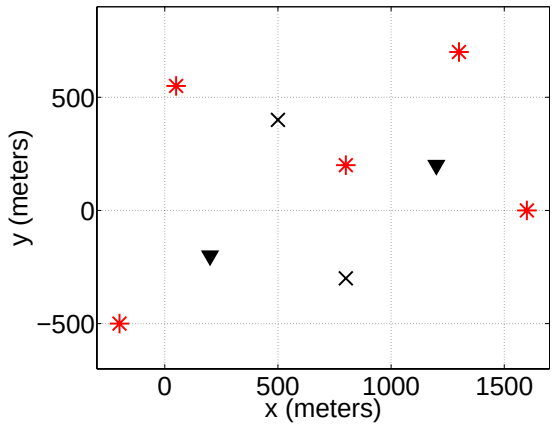
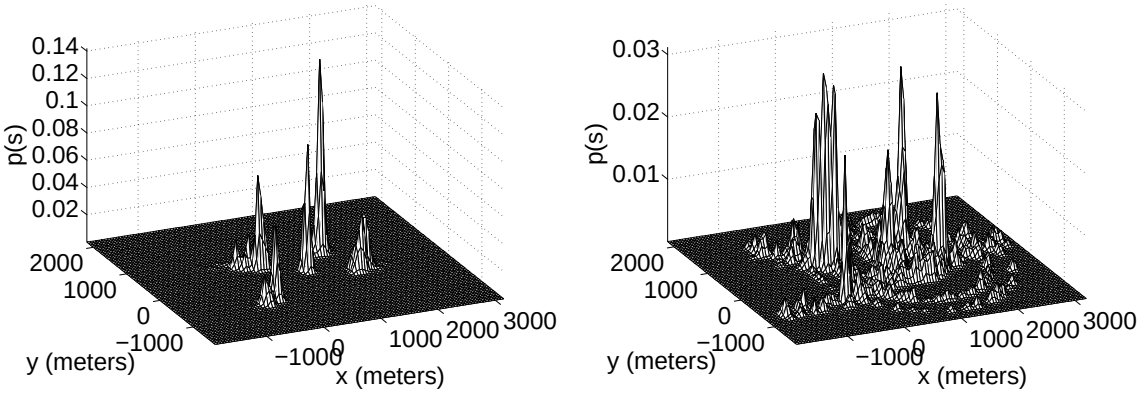


Fig. 8. Simulation setup for 5 targets: ▼ represent range-Doppler sensors, × represent bearing sensors and ∗ represent targets.

In this simulation, not all nodes are capable of observing both targets; 4 nodes observe both targets while the remaining 6 nodes are capable of observing only one of the two targets. It can be seen in Figure 11 that the posterior distribution is multimodal and the dominant peaks occur at the true target states.



(a) Low complexity. (b) Low latency.

Fig. 9. Simulation example comparing the two initialization algorithms for 5 targets.

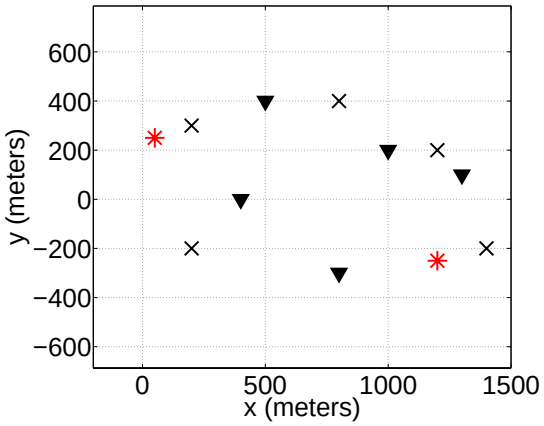
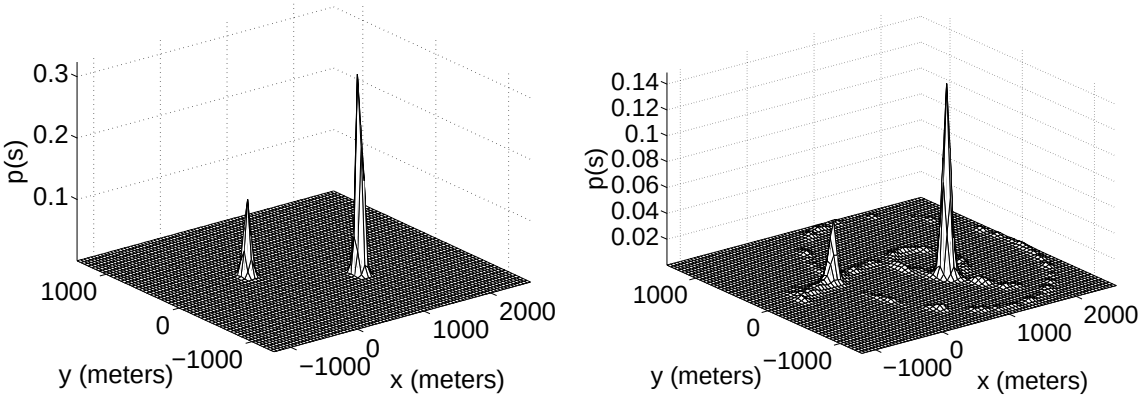


Fig. 10. Sensor and target positions in a large network: ▼ represent range-Doppler sensors, × represent bearing sensors and ∗ represent targets.



(a) Low complexity. (b) Low latency.

Fig. 11. Simulation example comparing the two initialization algorithms for a large network with 10 sensor nodes.

VIII. Conclusions

Two algorithms for generating the initial probability distribution are proposed for multiple targets in a distributed heterogeneous smart sensor network: a low complexity algorithm that requires three passes through the network for global initialization, and a low latency algorithm that requires only two passes. Our algorithms take into account missing data and clutter in the measurements available at the various sensor nodes. Monte Carlo methods are used to sequentially sample the state space to generate particles and robust weighting functions are used to represent the degree of belief in each particle. The final state vector distribution is represented using this weighted set of particles. This set of weighted particles can be used to make various inferences about the targets' states and also to initialize various distributed tracking algorithms.

The low latency algorithm reduces the communication load at the expense of increased computation at each node: $O(D^2)$ operations at each node for the low latency algorithm compared to $O(D)$ operations for the low complexity algorithm, where D represents the number of particles used to represent the posterior distribution of interest.

In this paper, we assume a fixed one hop sequential communication path from the first node to the last node in the network. In a real world implementation, this simplified communication protocol can be replaced by more efficient protocols with minor modifications to the proposed algorithms. The performance of the algorithms will not be affected as long as every node provides its input to the network at the proposal and weighting stages at most one time.

REFERENCES

[1] J. Manyika and H. Durrant-Whyte, *Data Fusion and Sensor Management: A Decentralized Information-Theoretic Approach*, Prentice Hall, 1994.

[2] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus, "Tracking a moving object with a binary sensor network," in *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, Los Angeles, California, 5-7 November 2003, pp. 150–161.

[3] R. Snelick, U. Uludag, A. Mink, M. Indovina, and A. Jain, "Large scale evaluation of multimodal biometric authentication using state-of-the-art systems," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, pp. 450–455, 2005.

[4] A. Dogandzic and B. Zhang, "Distributed estimation and detection for sensor networks using hidden markov random field models," *IEEE Transactions on Signal Processing*, vol. 54, no. 8, pp. 3200–3215, 2006.

[5] J. Yedidia, W.T. Freeman, and Y. Weiss, "Generalized belief propagation," *Advances in Neural Information Processing Systems*, vol. 13, pp. 689–695, 2001.

[6] R. Chellappa and A. Jain, Eds., *Markov random fields. Theory and application*, Boston: Academic Press, 1993.

[7] E.B. Sudderth, A.T. Ihler, W.T. Freeman, and A.S. Willsky, "Nonparametric belief propagation," in *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Madison, Wisconsin, 18-20 June 2003, vol. 1, pp. 605–612.

[8] M. Isard, "PAMPAS: real-valued graphical models for computer vision," in *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Madison, Wisconsin, 18-20 June 2003, vol. 1, pp. 613–620.

[9] A.T. Ihler, E.B. Sudderth, W.T. Freeman, and A.S. Willsky, "Efficient multiscale sampling from products of Gaussian mixtures," in *Neural Information Processing Systems 17*, Vancouver, British Columbia, Canada, 9-11 December 2003.

[10] A.T. Ihler, J.W. Fisher, R.L. Moses, and A.S. Willsky, "Nonparametric belief propagation for self-localization of sensor networks," *IEEE Journal on Selected Areas in Communication*, vol. 23, no. 4, pp. 809–819, April 2005.

[11] M.J. Coates, "Distributed particle filtering for sensor networks," in *Third International Symposium on Information Processing in Sensor Networks*, Berkeley, California, 26-27 April 2004, pp. 99–107.

[12] I. Leichter, M. Lindenbaum, and E. Rivlin, "A probabilistic framework for combining tracking algorithms," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Washington, DC, June 27–July 2 2004, vol. 2, pp. 445–451.

[13] M. Borkar, V. Cevher, and J. H. McClellan, "Estimating target state distributions in a distributed sensor network using a Monte-Carlo approach," in *2005 IEEE Workshop on Machine Learning for Signal Processing*, Mystic, Connecticut, 28-30 Sept. 2005, pp. 305–310.

[14] M. Borkar, V. Cevher, and J. H. McClellan, "A Monte-Carlo method for initializing distributed tracking algorithms with acoustic propagation delay compensation," to appear in *Journal of VLSI Signal Processing Systems*, Invited paper, also available at http://www.umiacs.umd.edu/users/volkan/JVLSIMilind.pdf.

[15] V. Cevher, R. Chellappa, F. Shah, R. Velmurugan, and J. H. McClellan, "An acoustic multi-target tracking system using random sampling consensus," to appear in *Proceedings of the 2007 IEEE Aerospace Conference*, Big Sky, Montana, 3–10 March 2007.

[16] V. Cevher, R. Velmurugan, and J. H. McClellan, "A range-only multiple target particle filter tracker," in *2006 IEEE International Conference on Acoustics, Speech and Signal Processing*, Toulouse, France, May 2006, vol. 4, pp. 905–908.

[17] E. Brookner, *Tracking and Kalman Filtering Made Easy*, John Wiley and Sons, 1988.

[18] P. R. Kalata and K. M. Murphy, "$\alpha - \beta$ target tracking with track rate variations," in *IEEE Proceedings of the Twenty-Ninth Southeastern Symposium on Systems Theory*, Cookeville, Tennessee, 9-11 March 1997, pp. 70–74.

[19] A. Doucet, "On sequential simulation-based methods for Bayesian filtering," Tech. Rep. CUED/F-INFENG/TR.310, Department of Engineering, University of Cambridge, 2001.

[20] J.M. Hammersley and D.C. Handscomb, *Monte Carlo methods*, Methuen, 1964.

[21] J.R. Munkres, *Analysis on Manifolds*, Perseus Books, 1990.

[22] Y. Bar-Shalom and T. Fortmann, *Tracking and Data Association*, Academic-Press, 1988.

[23] M. Isard and A. Blake, "Condensation – conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, no. 1, pp. 5–28, 1998.

[24] E. Parzen, "On estimation of a probability density function and mode," *Annals of Mathematical Statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

APPENDIX I
LOW COMPLEXITY INITIALIZATION ALGORITHM

- **Variables:**

  $\mathbf{s}_t^{(i)}$ = particle i at time t

  $w_t^{(i)}$ = weight of particle $\mathbf{s}_t^{(i)}$

  $\mathbf{z}_{m,t}$ = the measurement from node m at time t

  $\mathbf{z}_t = \{\mathbf{z}_{1,t}, \ldots, \mathbf{z}_{M,t}\}$

  D = Number of particles used for initialization

  M = the total number of nodes

  n_s = number of nodes that provided input to the algorithm

  $w_{\text{num}}^{(i)}$ = numerator of the weights

  $w_{\text{den}}^{(i)}$ = denominator of the weights

- **Forward Pass: Sequentially Generate Particles**

  n_s = 0

  Node 1:

  – If there is a detection

    ∗ Generate D particles:

      · $\mathbf{s}_t^{(i)} \sim p\left(\mathbf{s}_t|\mathbf{z}_{1,t}\right),\ i = 1, \ldots, D$

    ∗ Each particle will have equal weight

    ∗ $n_s = 1$

  – Else

    ∗ $\mathbf{s}_t^{(i)} = 0,\ i = 1, \ldots, D$

  – Send $\{\mathbf{s}_t^{(i)}\}_{i=1}^{D}$ and n_s to Node 2.

  For *Node m*, $m = 2, \ldots M$:

  – Receive $\{\mathbf{s}_t^{(i)}\}_{i=1}^{D}$ and n_s from Node $(m-1)$.

  – Give each received particle a weight of n_s

  – If there is a detection

    ∗ Label the received particles $\{\mathbf{s}_{r,t}^{(i)}\}_{i=1}^{D}$.

    ∗ Generate D new particles:

      · $\mathbf{s}_{n,t}^{(i)} \sim p\left(\mathbf{s}_t|\mathbf{z}_{m,t}\right),\ i = 1, \ldots, D$

    ∗ Each new particle will have equal weight

· Give each new particle a weight of 1

∗ From the $2D$ particles, obtain D particles $\{\mathbf{s}_t^{(i)}\}_{i=1}^{D}$ by using a weighted sampling with replacement.

∗ Each particle will now have equal weight

∗ $n_s = n_s + 1$

– If $m < M$

∗ Send $\{\mathbf{s}_t^{(i)}\}_{i=1}^{D}$ and n_s to Node $m+1$.

- **Reverse Pass: Disseminate Particles and Sequentially Generate Weights**

  For *Node m,* $m = M, \ldots, 1$:

  – If $m < M$

  ∗ Accept $\{\mathbf{s}_t^{(i)},\ w_{\text{num}}^{(i)},\ w_{\text{den}}^{(i)}\}_{i=1}^{D}$ from Node $(m+1)$.

  – Else

  ∗ $w_{\text{num}}^{(i)} = 1,\ i = 1, \ldots, D$

  ∗ $w_{\text{den}}^{(i)} = 0,\ i = 1, \ldots, D$

  – For $i = 1, ..., D$

  ∗ $w_{\text{num}}^{(i)} = w_{\text{num}}^{(i)} \cdot p(\mathbf{z}_{m,t}|\mathbf{s}_t^{(i)})$

  ∗ $w_{\text{den}}^{(i)} = w_{\text{den}}^{(i)} + \frac{p(\mathbf{z}_{m,t}|\mathbf{s}_t^{(i)})}{p(\mathbf{z}_{m,t})}$

  – Send $\{\mathbf{s}_t^{(i)},\ w_{\text{num}}^{(i)},\ w_{\text{den}}^{(i)}\}_{i=1}^{D}$ to Node $(m-1)$.

- **Forward Pass: Disseminate Weights**

  Node 1:

  – For $i = 1, ..., D$

  ∗ $w_t^{(i)} = \frac{w_{\text{num}}^{(i)}}{w_{\text{den}}^{(i)}}$

  – Normalize weights

  ∗ $w_t^{(i)} = \frac{w_t^{(i)}}{\sum_{i=1}^{D} w_t^{(i)}},\ i = 1, \ldots, D$

  – Send $\{w_t^{(i)}\}_{i=1}^{D}$ to Node 2.

  For *Node m,* $m = 2, \ldots M$:

  – Accept $\{w_t^{(i)}\}_{i=1}^{D}$ from Node $(m-1)$.

  – If $m < M$

  ∗ Send $\{w_t^{(i)}\}_{i=1}^{D}$ to Node $(m+1)$.

APPENDIX II
LOW LATENCY INITIALIZATION ALGORITHM

- **Variables:**

  $\mathbf{s}_t^{(i)}$ = particle i at time t

  $w_t^{(i)}$ = weight of particle $\mathbf{s}_t^{(i)}$

  $\mathbf{z}_{m,t}$ = the measurement from node m at time t

  $\mathbf{z}_t = \{\mathbf{z}_{1,t}, \ldots, \mathbf{z}_{M,t}\}$

  D = Number of particles used for initialization

  M = the total number of nodes

  n_s = number of nodes that provided input to the algorithm

  $W(\cdot)$ = Parzen window kernel

- **Forward Pass: Sequentially Generate Particles and Weights**

  $n_s = 0$

  Node 1:

  - If there is a detection,
    - Generate D particles:
      - $\mathbf{s}_t^{(i)} \sim p(\mathbf{s}_t|\mathbf{z}_{1,t}), i = 1, \ldots, D$
    - Assign Weights:
      - $w_t^{(i)} = \frac{1}{D},\ i = 1, \ldots, D$
    - $n_s = 1$
  - Else,
    - $\mathbf{s}_t^{(i)} = 0,\ i = 1, \ldots, D$
    - $w_t^{(i)} = 0,\ i = 1, \ldots, D$
  - Send $\{\mathbf{s}_t^{(i)},\ w_t^{(i)}\}_{i=1}^{D}$ and n_s to Node 2.

  For *Node m*, $m = 2, \ldots M$:

  - Receive $\{\mathbf{s}_t^{(i)},\ w_t^{(i)}\}_{i=1}^{D}$ and n_s from Node $(m-1)$.
    - These particles are distributed according to $\sum_{\tilde{m}=1}^{n_s} p(\mathbf{s}_t|\mathbf{z}_{\tilde{m},t})$.
    - Particles and weights together represent $p(\mathbf{s}_t|\mathbf{z}_{1,t}, \mathbf{z}_{2,t}, \ldots, \mathbf{z}_{n_s,t}) \propto \prod_{\tilde{m}=1}^{n_s} p(\mathbf{s}_t|\mathbf{z}_{\tilde{m},t})$
  - If there is a detection
    - Label the received particles and weights $\{\mathbf{s}_{r,t}^{(i)},\ w_{r,t}^{(i)}\}_{i=1}^{D}$.
    - Generate D new particles:

- $\mathbf{s}_{n,t}^{(i)} \sim p\left(\mathbf{s}_t|\mathbf{z}_{m,t}\right), i = 1, \ldots, D$

* For $i = 1, \ldots, D$, determine scaled weights:
  - $\tilde{w}_{n,t}^{(i)} = p\left(\mathbf{s}_{n,t}^{(i)}|\mathbf{z}_{m,t}\right) \cdot \sum_{j=1}^{D} w_{r,t}^{(j)} W\left(\mathbf{s}_{n,t}^{(i)} - \mathbf{s}_{r,t}^{(j)}\right)$
  - $\tilde{w}_{r,t}^{(i)} = p\left(\mathbf{s}_{r,t}^{(i)}|\mathbf{z}_{m,t}\right) \cdot \sum_{j=1}^{D} w_{r,t}^{(j)} W\left(\mathbf{s}_{r,t}^{(i)} - \mathbf{s}_{r,t}^{(j)}\right)$
* From the $2D$ pairs of particles and scaled weights, sample D pairs as follows:
  - Assign weighting $\hat{w} = n_s$ to $\{\mathbf{s}_{r,t}^{(i)},\ \tilde{w}_{r,t}^{(i)}\}_{i=1}^{D}$.
  - Assign weighting $\hat{w} = 1$ to $\{\mathbf{s}_{n,t}^{(i)},\ \tilde{w}_{n,t}^{(i)}\}_{i=1}^{D}$.
  - Perform weighted sampling with replacement according to weights given by $\hat{w}$ to generate the set $\{\mathbf{s}_t^{(i)},\ \tilde{w}_t^{(i)}\}_{i=1}^{D}$
* For $i = 1, \ldots, D$, modify the weights to account for the particle support:
  - $w_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^{D} W\left(\mathbf{s}_t^{(i)} - \mathbf{s}_t^{(j)}\right)}$
* Normalize Weights:
  - $w_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^{D} w_t^{(j)}}, i = 1, \ldots, D$
* Final set of particles are distributed according to $\sum_{\tilde{m}=1}^{m} p\left(\mathbf{s}_t^{(i)}|\mathbf{z}_{\tilde{m},t}\right)$.
* Final particles and weights together represent $p\left(\mathbf{s}_t^{(i)}|\mathbf{z}_{1,t}, \mathbf{z}_{2,t}, \ldots, \mathbf{z}_{m,t}\right) \propto \prod_{\tilde{m}=1}^{m} p\left(\mathbf{s}_t^{(i)}|\mathbf{z}_{\tilde{m},t}\right)$.
* $n_s = n_s + 1$

– If $m < M$

* Send $\{\mathbf{s}_t^{(i)},\ w_t^{(i)}\}_{i=1}^{D}$ and n_s to Node $m + 1$.

- **Reverse Pass: Disseminate Particles and Weights**

Node M:

– Send $\{\mathbf{s}_t^{(i)},\ w_t^{(i)}\}_{i=1}^{D}$ to Node $M - 1$.

For *Node m*, $m = M - 1, \ldots, 1$

– Receive $\{\mathbf{s}_t^{(i)},\ w_t^{(i)}\}_{i=1}^{D}$ from Node $m + 1$.

– if $m > 1$

* Send $\{\mathbf{s}_t^{(i)},\ w_t^{(i)}\}_{i=1}^{D}$ to Node $m - 1$.