Decentralized State Initialization with Delay Compensation for Multi-Modal Sensor Networks

Milind Borkar, *Student Member, IEEE,* Volkan Cevher, *Member, IEEE,*
James H. McClellan, *Fellow, IEEE*

Abstract

Decentralized processing algorithms are attractive alternatives to centralized algorithms for target tracking applications in smart sensor networks since they provide the ability to scale, reduce vulnerability, reduce communication, and share processing responsibilities among individual nodes. Sharing the processing responsibilities allows parallel processing of raw data at the individual nodes. However, this introduces other difficulties in multi-modal smart sensor networks, such as non-observability of the targets' states at any individual node and various delays such as varying processing delays, communication delays and signal propagation delays for the different modalities. In this paper, we provide a novel algorithm to determine the initial probability distribution of multiple targets' states in a decentralized manner. The targets' state vectors consist of the targets' positions and velocities on the 2D plane. Our approach can determine the state vector distribution even if the individual sensors alone are not capable of observing it. Our approach can also compensate for varying delays among the assorted modalities. The resulting distribution can be used to initialize various tracking algorithms. Our approach is based on Monte-Carlo methods, where the state distributions are represented as a weighted set of discrete state realizations. A robust weighting strategy is formulated to account for missed detections, clutter and estimation delays. To demonstrate the effectiveness of the algorithm, we simulate a network with direction-of-arrival nodes and range-doppler nodes.

I. Introduction

In sensor networks, the data available at the outputs of individual sensing elements can be processed in either a centralized or a decentralized fashion. Centralized processing is characterized by a single central processor. Raw data from all the sensors is transmitted to the central processor, which has the responsibility of processing all the incoming data and producing meaningful outputs. On the other hand, in a decentralized system, multiple processors exist and the data processing responsibilities are split among these processors. In the extreme case, each sensor may have a dedicated processor. These processors operate on the raw sensor data and the outputs are then fused in a decentralized manner. Block diagrams representing centralized and decentralized processing are given in Figure 1.

In sensor networks, decentralized 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

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.

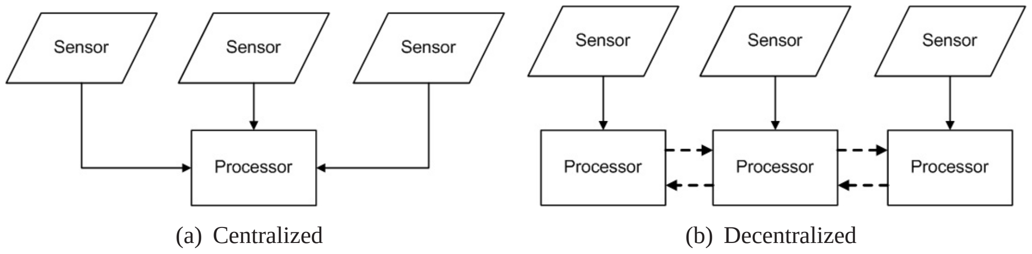


Fig. 1. Centralized vs. decentralized processing. The solid lines represent raw data whereas the dashed lines represent sufficient statistics.

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. To overcome these drawbacks, a decentralized processing approach is more attractive.

Decentralized processing stipulates processing capabilities at individual sensors. We denote a sensor that has the ability to process data and communicate with neighboring sensors in addition to sensing the environment as a *smart sensor*. Decentralized processing eliminates the need for a central processing node. Since a smart sensor can process its own data, it only needs to transmit sufficient statistics in the communication channel, minimizing the communication among sensors. Communication consumes more battery power than computation [2], hence smart sensor networks with decentralized processing have reduced power requirements.

Processing data in multi-modal smart sensor networks to generate global estimates is a challenging problem. One of the issues is data fusion in an efficient and effective manner. The decentralized data fusion (DDF) methods given in [1] are highly effective in networks with sensors sharing the same modality. In the case of multi-modal networks, it is rather difficult to analytically fuse data and represent global knowledge in a decentralized manner. This difficulty increases when there are multiple targets. State observability is another issue since each individual sensor may only be able to observe a limited subspace of the target state. Another issue arises when the different modalities have varying signal propagation velocities since some subset of sensor nodes, e.g. acoustic sensors, may contribute delayed information about the target state, hence leading to biased estimates of the targets' current states. This effect is magnified when the targets are moving with high velocities, are at a large range from the sensing node or are maneuvering. Time delays may also be introduced in the system by processing and communication latencies and these may be different at different nodes.

In this paper, a novel method for multiple targets' state initialization in multi-modal smart sensor networks is proposed in a decentralized framework. Our algorithm addresses issues related to data fusion, observability, and varying time delays for the multi modal sensor nodes. Monte-Carlo methods are used to generate discrete approximations to the targets' state distributions. These distributions are represented using hypothesized state vectors called particles and their associated weights. The output of the initialization algorithm can be used to initialize various decentralized joint tracking (DJT) algorithms such as those described in [3]–[8].

Our algorithm satisfies the typical constraints of a decentralized system. The communication between individual sensors has a fixed bandwidth. Since the data propagated between sensors is the cumulative state information, the amount of data passed between individual sensors does not increase with the number of sensors. The sensor types focused on are direction-of-arrival (DOA) nodes (e.g., acoustic arrays with known microphone positions) and range-doppler nodes (e.g., radar sensors). However, the results are general and can be extended to networks using other sensor modalities. Each smart sensor has a built-in organic pre-processor that runs a tracking algorithm operating in a state space that is specifically determined by that sensor modality. We refer to the tracking algorithms running at the individual sensors as organic trackers. The DJT operates in a state space that may be different from the state spaces of the organic trackers at the individual nodes. We assume that each organic tracker is capable of detecting a new target. When an organic tracker detects a new target in its limited subspace, it transmits information throughout the network to generate the target's state distribution. We use a robust weighting strategy that can accommodate clutter, missing data and delays due to signal propagation, processing and communication. Moreover, communication takes place between neighboring sensors only and we assume that there is a predefined path for the information flow through the network from the first sensor to the last sensor. However, the algorithm is still applicable to networks with other communication strategies.

The organization of the paper is as follows. Section II gives a brief overview of the overall system design. Section III introduces the acoustic and radar trackers. Section IV proposes our Monte-Carlo approach for the decentralized estimation of the targets' state distribution. Section V focuses on communication between the nodes in the network. Section VI proposes modifications to the initialization algorithm of Section IV to compensate for data delays that may be present in the system. Section VII gives a brief overview of a decentralized multi-modal joint tracking algorithm with delay compensation. Section VIII demonstrates the effectiveness of the proposed algorithms on synthetic data. Conclusions and future work follow in Section IX.

II. System Design

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. This block could perform beamforming (for acoustic nodes), radar pre-processing and batch processing of measurements to generate motion estimates. These organic state estimates are used to provide input to the DJT block that tracks targets in a global state space which may be different from the organic state space. The organic state estimates are also used to maintain target tracks within the sensor node in the organic tracker block. This is important since it allows the detection of a new target. When a target that does not correspond to existing target tracks is detected, the organic state estimates are fed into the decentralized initialization block. The decentralized initialization block takes in organic state estimates for new targets from multiple nodes and combines them to produce state estimates in the global state space used by the DJT.

This paper focuses on the decentralized initialization block. Some of the ideas developed for the initialization algorithm are then extended to the DJT.

III. Organic Sensors and Pre-Processors

The two types of sensor nodes used to demonstrate the proposed algorithms are DOA sensors and range-doppler sensors. The organic DOA trackers 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. The organic range-doppler trackers operate in the

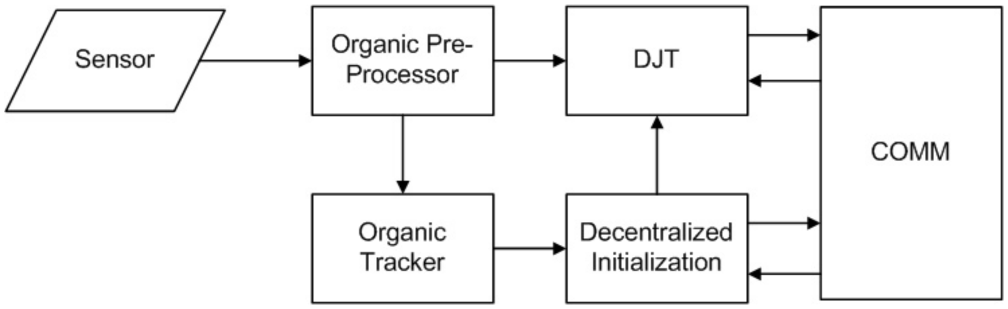


Fig. 2. System block diagram of a smart sensor node.

$[r\ v_r]^T$ space where r is the range to the target and v_r is the target's radial velocity. Detailed descriptions about these trackers can be found in [9]–[17].

The focus of this paper is to generate probability distributions representing multiple targets' states in the $[x\ y\ v_x\ v_y]^T$ space where x and y are the Cartesian coordinates of a target's location, and v_x and v_y are the velocity components along the x-y directions. The probability distributions generated at the output of the initialization algorithm are used as the initial estimates of the state vector distributions by the DJT. Notice that the true location and velocity of any target is not observable at any of the individual nodes and that the organic trackers operate in different state spaces that have lower dimensionality than the DJT state space. This means there is a many to one mapping from the state space used by the organic trackers to the DJT state space. It is assumed that organic trackers are running at the different nodes and the outputs of the organic trackers are used to generate the desired probability distribution. The sensor network is assumed to be calibrated so that each sensor is aware of its own location. However, sensors need not be aware of the locations of other sensors in the network.

IV. A Monte-Carlo Approach for the Decentralized Estimation of the Targets' Initial State Distribution

The initialization algorithm uses a novel Monte-Carlo approach to generate an approximation to the state vector distributions using a weighted set of particles. To generate an optimal particle distribution that minimizes the variance of the particle weights, one must sample from the true posterior distribution [18]. Using Bayes' rule, the posterior distribution can be expressed as

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

where $\mathbf{x}_t$ represents the target state vector at time t, and $\mathbf{z}_t$ is the vector of measurements from all M sensors at time t. We assume that the measurements at the individual nodes are independent, conditioned on the current state. Hence, the combined data likelihood for all sensors can be factored into the product of the data likelihoods at the individual sensor nodes. Assuming no prior knowledge of the true targets' state distributions, $p(\mathbf{x}_t)$ is chosen to be uniform and is dropped from the equation. The distribution $p(\mathbf{z}_t)$ is simply a proportionality constant since it does not depend on the state. Therefore, (1) can be simplified as

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

where $\mathbf{z}_{m,t}$ is the set of measurements from the m^{th} sensor at time t. We choose not to communicate raw data between nodes to limit communication bandwidth. Thus, determining the posterior distribution analytically is impossible. Instead we chose, as our proposal function,

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

which is an equally weighted mixture of the individual posterior distributions from the different nodes. It is desired that the targets' should be initialized in the $\mathbf{x}_t = [x_t \ y_t \ v_{x_t} \ v_{y_t}]^T$ space. Assume the state vector is n-dimensional in general. Also, assume that at sensor m, target measurements 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 will not contribute any useful information for tracking, and can be discarded. Thus, without loss of generality, we can assume that each feature is a function of the state vector.

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

We now determine the conditions that $f_m(\cdot)$ must satisfy to be able to sample from (3). Let $f_m(\cdot)$ be a continuously differentiable vector valued function. If and only if all features in the feature vector at a particular sensor provide complimentary information without redundancy, then

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

If any of the features provide redundant information, those particular features can be discarded to give a feature space of reduced dimension and no redundancy. Therefore, we can assume all features provide complementary information and (5) is satisfied.

First, consider the case, when $s < n$. Given the features $\hat{\mathbf{z}}_{m,t}$, the system in (4) is underdetermined. Therefore, there exist infinitely many solutions for $\mathbf{x}_t$. These solutions form a level set in the state space. In some cases, the level set can be represented by explicit equations relating the state variables. However, this may not be possible in most cases even though the level sets do exist. Let α be any solution of (4). By the implicit function theorem [19], in the neighborhood of α, the level set $L_f(\mathbf{z}_{m,t})$ 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 and adding appropriate noise determined by the measurement model.

Now consider the case, when $s = n$. By the inverse function theorem [19], 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

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

Hence, particles can be sampled from an appropriate distribution centered at $\mathbf{x}_t$ with variance determined by the measurement model.

Since measurements $\mathbf{z}_{m,t}$ are available as random realizations of the features, we propose to use the measurement vectors as estimates of the true feature vectors in the implementation of the preceding procedure. In this manner, particles can be sampled from the individual posterior distributions without sharing raw data.

If sensor m is a binary detection sensor (i.e., the sensor's output is binary depending on whether a target is detected or not) then the mapping from the target state space to the feature space is not differentiable. However, this is a special case since the output is a binary function on some $f_m(\cdot)$. In this situation, all points in the domain of $f_m(\cdot)$ that would result in a detection are possible target states and can be denoted by a set S_d. The same rules for sampling explained above can be applied to points in S_d and the procedure remains unchanged.

In the case of DOA nodes and range-doppler nodes, particles can be sampled from the individual posterior distributions as follows:

For DOA nodes:

$$r^{(i)} \sim U[0, r_{\max}) \tag{7}$$

$$\theta^{(i)} \sim N(\theta_{m,t}, \sigma_{\theta_{m,t}}) \tag{8}$$

$$Q^{(i)} \sim N(Q_{m,t}, \sigma_{Q_{m,t}}) \tag{9}$$

$$\phi^{(i)} \sim N(\phi_{m,t}, \sigma_{\phi_{m,t}}) \tag{10}$$

$$x_t^{(i)} = r^{(i)} \cos(\theta^{(i)}) + s_{m,x} \tag{11}$$

$$y_t^{(i)} = r^{(i)} \sin(\theta^{(i)}) + s_{m,y} \tag{12}$$

$$v_{x_t}^{(i)} = e^{Q^{(i)}} r^{(i)} \cos(\phi^{(i)}) \tag{13}$$

$$v_{y_t}^{(i)} = e^{Q^{(i)}} r^{(i)} \sin(\phi^{(i)}) \tag{14}$$

For Range-Doppler nodes:

$$r^{(i)} \sim N(r_{m,t}, \sigma_{r_{m,t}}) \tag{15}$$

$$\theta^{(i)} \sim U[0, 2\pi) \tag{16}$$

$$v_r^{(i)} \sim N(v_{r_{m,t}}, \sigma_{v_{r_{m,t}}}) \tag{17}$$

$$v_t^{(i)} \sim U(-(v_{\max}^2 - (v_r^{(i)})^2)^{0.5}, (v_{\max}^2 - (v_r^{(i)})^2)^{0.5}) \tag{18}$$

$$x_t^{(i)} = r^{(i)} \cos(\theta^{(i)}) \tag{19}$$

$$y_t^{(i)} = r^{(i)} \sin(\theta^{(i)}) \tag{20}$$

$$v_{x_t}^{(i)} = v_r^{(i)} \cos(\theta^{(i)}) + v_t^{(i)} \sin(\theta^{(i)}) \tag{21}$$

$$v_{y_t}^{(i)} = v_r^{(i)} \sin(\theta^{(i)}) - v_t^{(i)} \cos(\theta^{(i)}) \tag{22}$$

The m^{th} sensor position in Cartesian coordinates is given by $(s_{m,x}, s_{m,y})$. 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. Every DOA node has a range ambiguity, while every range-doppler node has a DOA ambiguity and a tangential velocity ambiguity. Therefore, 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 considered positive if the target is moving away from the node. Tangential velocity is considered positive if the tangential component points in the counterclockwise direction.

Using (7)-(22), one can sample particles from the individual posteriors. If the total number of nodes is M, then to sample D particles from the mixture given by (3), one can 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 are spread uniformly over the entire state space for that node and these particles do not add any information to the system. Instead of sampling these particles uniformly, it is more informative to sample only from the posteriors for the nodes that have detections. Hence, more particles cover the state space of interest. These disadvantages can be eliminated by following Step 1 of the algorithm in Appendix I, where a weighted resampling operation ensures that the individual posteriors for nodes with detections are equally weighted irrespective of the total number of nodes. Resampling does not require synchronization of the nodes.

Once the particles are sampled, they 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.

Using the results of [18], the particle weights for our problem are given by

$$w_t^{(i)} = \frac{p(\mathbf{x}_t^{(i)}|\mathbf{z}_t)}{\pi(\mathbf{x}_t^{(i)}|\mathbf{z}_t)}. \tag{23}$$

From (2) and (3), (23) can be simplified as follows:

$$w_t^{(i)} \propto \frac{\prod_{m=1}^{M} p(\mathbf{z}_{m,t}|\mathbf{x}_t^{(i)})}{\sum_{m=1}^{M} p(\mathbf{x}_t^{(i)}|\mathbf{z}_{m,t})}. \tag{24}$$

Using the Bayes' rule, we obtain the following expression for the posterior

$$p(\mathbf{x}_t^{(i)}|\mathbf{z}_{m,t}) = \frac{p(\mathbf{z}_{m,t}|\mathbf{x}_t^{(i)})p(\mathbf{x}_t^{(i)})}{p(\mathbf{z}_{m,t})}. \tag{25}$$

Since no prior information about the state vector is available, $p(\mathbf{x}_t)$ is assumed uniform over its natural space and is dropped from the equation. Thus, (24) simplifies to

$$w_t^{(i)} \propto \frac{\prod_{m=1}^{M} p(\mathbf{z}_{m,t}|\mathbf{x}_t^{(i)})}{\sum_{m=1}^{M} \frac{p(\mathbf{z}_{m,t}|\mathbf{x}_t^{(i)})}{p(\mathbf{z}_{m,t})}}. \tag{26}$$

Thus, the weights for the particles can be calculated, up to a proportionality, by evaluating a quotient in which the numerator is the product of the data likelihoods from the different nodes and the denominator is the weighted sum of the same likelihoods. Hence, cumulative update of the weights can be accomplished, if the numerators and denominators are both communicated between nodes.

When the final particles are proposed, there is an ambiguity as to which sensor proposed a particular particle. If a particular sensor has multiple detections, then this brings in additional complexity, since the particles can not be associated with their detectors. If a simple Gaussian likelihood function is used and the likelihood for a particle is zero at one of the sensors, then based on (26), its overall weight will also be zero. This situation occurs if even one sensor does not detect a target. In such situations, one would not want the overall weight of the particle to be zero since a target is present with high probability. To avoid this degeneracy, it is important that a robust likelihood function that accounts for missed detections is used.

The approach used here is similar to the approach used in [20], [21]. Assume that sensor m has K measurements. Then, given a particle $\mathbf{x}_t^{(i)}$, the measurements $\mathbf{z}_{m,k,t}$, $k = 1, ..., K$, 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 miss is set equal to a constant q. It is assumed that there is an equal probability for each of the K measurements to be a true measurement and the true target measurement is Gaussian distributed about the true target state. Thus, as shown in [20] the likelihood function can be simplified as:

$$p(\mathbf{z}_{m,t}|\mathbf{x}_t^{(i)}) \propto 1 + \frac{1-q}{\sqrt{(2\pi)^n|\Sigma|}q\lambda} \cdot \sum_{k=1}^{K} \exp\{-0.5(\mathbf{z}_{m,k,t} - f_m(\mathbf{x}_t^{(i)}))^T \mathbf{\Sigma}_{m,t}^{-1} (\mathbf{z}_{m,k,t} - f_m(\mathbf{x}_t^{(i)}))\}, \tag{27}$$

where n is the dimensionality of the measurement at sensor m and $\mathbf{\Sigma}_{m,t}$ is the covariance of the Gaussian distribution.

Steps 2 and 3 of the algorithm in Appendix I explain the weighting step. The set of particles along with their associated weights give a discrete representation of the probability distribution of the targets in the desired state space.

V. Communication

For the purpose of this paper, we assume a fixed sequential communication path from the first node to the last node. Using this communication protocol, the algorithm given in Appendix I requires three passes through the communication chain.

In the first pass, a varying set of a fixed number of N particles representing the combined information from all preceding nodes is transmitted through the communication chain. Along with the particles, a single number $\tilde{w}$ representing the number of sensors that detected a target is transmitted. At the end of the first pass, node M has the final set of N particles that represent particles proposed using (3). These need to be propagated back to all the other nodes so that weights can be computed.

In the second pass, the communication path is reversed. The final set of N particles are propagated back sequentially to node 1. It was shown that the individual components of the particle weights in (26) can be evaluated independently at each node and the numerator and denominator of the overall weights can be transmitted cumulatively. Thus the data communicated between pairs of nodes consists of the final set of N particles, the N numerator and N denominator components representing 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 and node 1 is the only node with the final set of weights.

In the third pass, the final set of weights are propagated throughout the network using the forward communication path. At the end of the third pass, all nodes have the same particles and weights.

In a real world implementation, the simplified communication protocol given in this paper can be replaced by more efficient protocols. The performance of the algorithm 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.

VI. Compensating for Delays due to Signal Propagation, Processing and Communication

Since the sensor network is multi-modal, various delays could be introduced into the system. There may be unequal processing latencies at each node, unequal communication delays for different pairs of nodes, and unequal signal propagation velocities for different modalities. For example, consider a network consisting of acoustic arrays (DOA nodes) and range-doppler nodes (radar). Assume for now that processing and communication is instantaneous. Hence the only delays introduced in the system are due to unequal signal propagation velocities. Electromagnetic waves travel at the speed of light. Hence, range and radial velocity estimates provided by the radar pre-processors can be assumed to represent the targets' current states. However, the acoustic signal propagates through the air at a much lower velocity. Thus, the signal received at the acoustic sensors at the current time may have been generated by the targets at a previous time and hence the state estimates available at the output of the acoustic pre-processors may represent previous targets' states. Using these estimates in the initialization or tracking algorithms will lead to erroneous state estimates. The effect increases if the targets are maneuvering, are at large ranges from the sensor node or are moving at high velocities. Hence, it is essential that the known delays in the system are compensated.

Assume that the modality at sensor m has signal propagation velocity v_m. If sensor m is an acoustic node, temporary particles $\tilde{\mathbf{x}}_t^{(i)} = [\tilde{x}_t^{(i)} \; \tilde{y}_t^{(i)} \; \tilde{v}_{xt}^{(i)} \; \tilde{v}_{yt}^{(i)}]^T$ are sampled as given in (7)-(14) using the current acoustic state estimates $\mathbf{z}_{m,t} = [\theta_{m,t} \; Q_{m,t} \; \phi_{m,t}]^T$. If sensor m is a radar node, the particles are sampled as given in (15)-(22) using the current state estimates $\mathbf{z}_{m,t} = [r_{m,t} \; v_{rm,t}]^T$. These particles represent the targets' state distributions at a previous time. Particles representing the current state vector are proposed using particles $\tilde{\mathbf{x}}_t^{(i)}$ and the state transition model. In this case, the state transition model is assumed to be a locally constant velocity model. To propose particles representing the targets' current state distribution, the total delay must be known. In the case of acoustic sensors, to determine the signal propagation delay, the targets' ranges must be known. Using the targets' ranges, the signal propagation delay can be estimated and the particles $\tilde{\mathbf{x}}_t^{(i)}$ can be propagated forward through the state transition model to produce particles $\mathbf{x}_t^{(i)}$ representing the state vector.

A target's range is not observable at the acoustic nodes. However, the proposed particles $\tilde{\mathbf{x}}_t^{(i)}$ represent hypothesized target states and each particle has a unique range from the sensor. Hence the time window for prediction is determined independently for each particle based on the particle's range. The final particles are proposed as follows

$$T^{(i)} = \frac{r^{(i)}}{v_m} + d_{\text{proc}} + d_{\text{comm}}, \tag{28}$$

$$\mathbf{x}_t^{(i)} \sim N(\mathbf{A}_{T^{(i)}}\tilde{\mathbf{x}}_t^{(i)}, {T^{(i)}}^2\mathbf{\Sigma}_{\mathbf{x}}), \tag{29}$$

where

$$\mathbf{A}_T = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{30}$$

and $\mathbf{\Sigma}_{\mathbf{x}}$ is the state transition noise covariance matrix in the Cartesian coordinate space, d_{proc} is the processing delay and d_{comm} is the communication hop delay. The noise added to these final particles accounts for possible maneuvers.

Once the particles are proposed, they need to be weighted. Weights are assigned using (26). However, the likelihood function $p(\mathbf{z}_{m,t}|\mathbf{x}_t^{(i)})$ needs to be modified to implement the delay compensation step introduced in this section. It can be shown that the appropriate likelihood function is given by

$$\begin{aligned} p(\mathbf{z}_{m,t}|\mathbf{x}_t^{(i)}) &\propto 1 + \frac{1-q}{\sqrt{(2\pi)^n|\widetilde{\mathbf{\Sigma}}_{m,T^{(i)}}|}q\lambda} \times \\ &\sum_{k=1}^{K} \exp\left(-\frac{1}{2}(h_{T^{(i)}}(\mathbf{z}_{m,k,t}) - g(\mathbf{x}_t^{(i)}))^H \widetilde{\mathbf{\Sigma}}_{m,T^{(i)}}^{-1} (h_{T^{(i)}}(\mathbf{z}_{m,k,t}) - g(\mathbf{x}_t^{(i)}))\right), \end{aligned} \tag{31}$$

$$\widetilde{\mathbf{\Sigma}}_{m,T^{(i)}} = \mathbf{J}(h_{T^{(i)}}(\mathbf{z}_{m,k,t}))\mathbf{\Sigma}_{m,t}\mathbf{J}^H(h_{T^{(i)}}(\mathbf{z}_{m,k,t})) + {T^{(i)}}^2\mathbf{\Sigma}_{m,s}, \tag{32}$$

where $\mathbf{\Sigma}_{m,s}$ is the state transition noise covariance matrix for the organic tracker at sensor m and $h_{T^{(i)}}(\mathbf{z}_{m,k,t})$ is the state transition vector for the organic tracker. For acoustic nodes, $h_{T^{(i)}}(\mathbf{z}_{m,k,t})$ is given by

$$h_T(\mathbf{z}_{m,k,t}) = \begin{bmatrix} h_{\theta,T}(\mathbf{z}_{m,k,t}) \\ h_{Q,T}(\mathbf{z}_{m,k,t}) \\ h_{\phi,T}(\mathbf{z}_{m,k,t}) \end{bmatrix}, \tag{33}$$

$$h_{\theta,T}(\mathbf{z}_{m,k,t}) = \tan^{-1}\left(\frac{\sin(\theta_{m,k,t}) + \exp(Q_{m,k,t})T\sin(\phi_{m,k,t})}{\cos(\theta_{m,k,t}) + \exp(Q_{m,k,t})T\cos(\phi_{m,k,t})}\right), \tag{34}$$

$$h_{Q,T}(\mathbf{z}_{m,k,t}) = Q_{m,k,t} - \frac{1}{2}\log\left(1 + 2T\exp(Q_{m,k,t})\cos(\theta_{m,k,t} - \phi_{m,k,t}) + T^2\exp(2Q_{m,k,t})\right), \tag{35}$$

$$h_{\phi,T}(\mathbf{z}_{m,k,t}) = \phi_{m,k,t}. \tag{36}$$

Analytical derivations of (33) through (36) can be found in [9], [13]. For radar nodes, $h_{T^{(i)}}(\mathbf{z}_{m,k,t})$ is given by

$$h_T(\mathbf{z}_{m,k,t}) = \begin{bmatrix} h_{r,T}(\mathbf{z}_{m,k,t}) \\ h_{v_r,T}(\mathbf{z}_{m,k,t}) \end{bmatrix}, \tag{37}$$

$$h_{r,T}(\mathbf{z}_{m,k,t}) = r_{m,k,t} + Tv_{rm,k,t}, \tag{38}$$

$$h_{v_r,T}(\mathbf{z}_{m,k,t}) = v_{rm,k,t}. \tag{39}$$

Note that the covariance for the data likelihood is given by $\widetilde{\mathbf{\Sigma}}_{m,T^{(i)}}$ and consists of two components. The first component depends on the measurement noise. Since future states are predicted using non-linear combinations of elements of $\mathbf{z}_{m,k,t}$ given by $h_T(\mathbf{z}_{m,k,t})$, independent noise from the components of $\mathbf{z}_{m,k,t}$ is accumulated based on the Jacobian of $h_T(\mathbf{z}_{m,k,t})$. The second component comes from the state transition noise and is introduced to account for the possibility that the targets are maneuvering.

VII. Decentralized Multi-Target Tracking

A decentralized joint tracker is implemented using an approach similar to the one given in [22]. The tracker is implemented using synchronized particle filters that run at each node in the smart sensor network. Synchronized particle filters maintain the same set of particles at each node using synchronized noise sources or noise tables. These trackers run at each node in addition to the organic tracking algorithms.

A bootstrap approach is used in which the particle proposal function is simply the state transition distribution and the weighting function is simply the measurement likelihood. By exploiting the conditional independence of the measurements given the targets' states, the decentralized tracker can be implemented by sharing only the particle weights in a cumulative manner. In the case of varying signal propagation velocities among the various sensor modalities, the likelihood function given in (31) is used to compensate for signal propagation delays.

VIII. Simulations

For the simulations, we assume that the signal processing and the inter-node communication are instantaneous. Hence, the only delays introduced in the system are signal propagation delays and compensation is only required at the acoustic nodes.

Assume that a target appears at an x-y location (50 m, 50 m) with velocity of 4 m/s in the x-direction and 4 m/s in the y-direction. There are a total of four sensors in the field. Two acoustic sensor nodes are located at (100 m, 40 m) and (350 m, 60 m), whereas two radar nodes are located at (200 m, 150 m) and (275 m, -50 m). Organic trackers at the four nodes detect this target and produce estimates in their own state spaces.

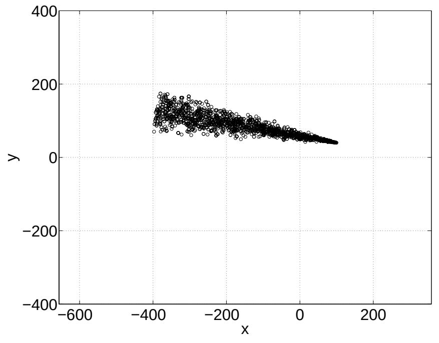
For this simulation, $D = 2000$ particles were used to sample the state space. Since the network is relatively small, and the target is moving slowly, the acoustic signal propagation delay compensation is not used in this simulation. To simulate the estimates available from the organic trackers (i.e., $[\theta\ Q\ \phi]^T$ from the DOA trackers and $[r\ v_r]^T$ from the range-doppler trackers), the measurements at each tracker are Gaussian distributed about the true values with standard deviations given by

$$\sigma_\theta = 2^o,\ \sigma_Q = 0.02\ \text{s}^{-1},\ \sigma_\phi = 8^o, \tag{40}$$

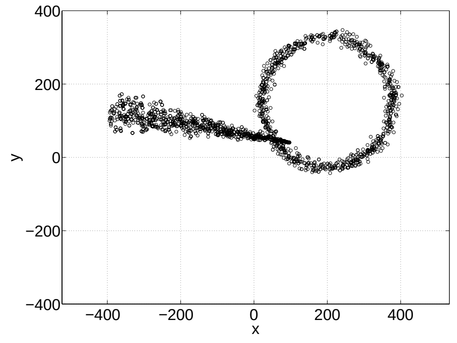
$$\sigma_r = 6\ \text{m},\ \sigma_{v_r} = 0.4\ \text{m/s}. \tag{41}$$

The clutter is modelled as a Poisson distributed random variable with parameter $\lambda = 1/7$.

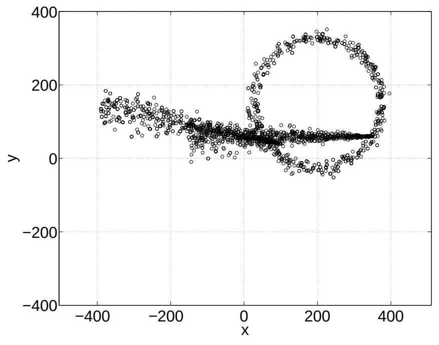
Figures 3(a) to 3(d) represent the sequential particle proposal stage of the algorithm. Although the state vector is four dimensional, the first four subfigures in Fig. 3 show only the x-y locations of the particles. In Fig. 3(a), sensor 1, which is a DOA sensor, detects the target at a particular angle and distributes 2000 particles along that angle up to an assumed maximum range. These particles are propagated to sensor 2, a range-doppler sensor. Sensor 2 receives the particles from sensor 1 and gives these particles a weight of 1 since they represent information from a single sensor. Sensor 2 detects the target at a particular range. Since bearing information is



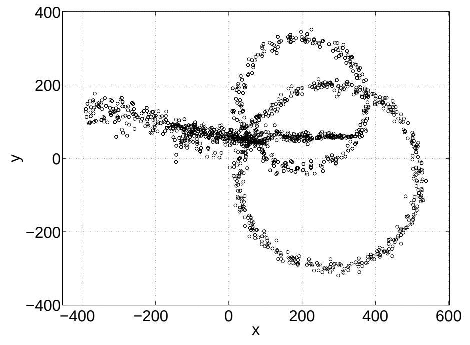
(a) Sensor 1



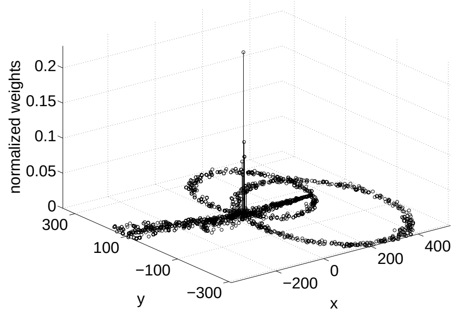
(b) Sensors 1,2



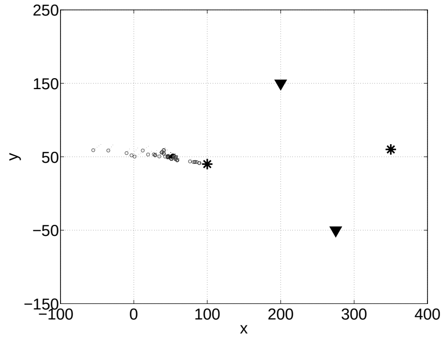
(c) Sensors 1,2,3



(d) Sensors 1,2,3,4



(e) Weighted Particles



(f) Final Particles

Fig. 3. Simulation example for initializing a single target.

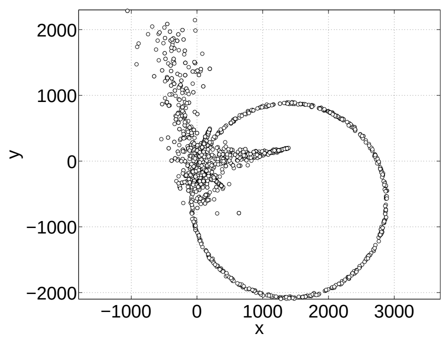
not available, sensor 2 distributes another 2000 particles about a circle with radius equal to the detected range and center at the sensor position. Out of the 4000 particles at sensor 2, 2000 particles are sampled uniformly with replacement. These particles are shown in Fig. 3(b), and are propagated to sensor 3, another DOA sensor. Sensor 3 receives the particles from sensor 2 and gives these particles a weight of 2 since these particles represent the combined information from two sensors. Sensor 3 detects the target at a particular angle and distributes another 2000 particles along that angle. These new particles have a weight of 1. From the 4000 particles at sensor 3, a weighted sampling with replacement is used to generate 2000 equally weighted particles. These particles are shown in Fig. 3(c) and are propagated to sensor 4, another range-doppler sensor. Sensor 4 receives the particles from sensor 3 and gives them a weight of 3 since they represent the combined information from 3 sensors. Then, sensor 4 detects the target at a particular range and distributes another 2000 particles along a circle with radius equal to the detection range and center at the sensor location. These new particles are given a weight of 1. From the 4000 particles at sensor 4, 2000 particles are obtained by using a weighted sampling with replacement. These final particles are plotted in Fig. 3(d) and are propagated back to all the sensors.

Weights are calculated for the final particles shown in Fig. 3(d). Particles along with their weights are shown in Fig. 3(e) and this represents the probability distribution of the target in the x-y space. As expected, the distribution is highly peaked about the true target state. Estimates of the true target state can be made based on this weighted set of particles. These estimates can be used to initialize any DJT.

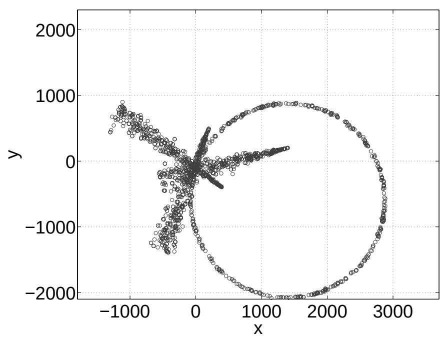
It is observed that the majority of particles have extremely low weights and do not contribute any useful information. To eliminate these particles and replicate those with high weights, the particles are sampled with replacement according to their weights to give the set of particles in Fig. 3(f). Here, the circles represent the particle positions and the lines, extending from the circles, represent the magnitude and the direction of the velocities. The stars represent the acoustic node locations and the triangles represent the radar locations. It can be seen that the final set of particles is concentrated around the true target state at $[50,\ 50,\ 4,\ 4]^T$ which is represented by the bold marker.

Even though the previous simulation worked in a small network with a slow moving target, signal propagation delay compensation is essential in large networks when targets have high velocities. This is illustrated in the following simulation.

A target is born at $\mathbf{x}_1 = [50,\ 0,\ 50,\ 50]^T$ and moves at an almost constant velocity. Three acoustic nodes are located at (400 m, -400 m), (200 m, 500 m) and (1400 m, 200 m) and a radar node is located at (1400 m, -600 m). The algorithm is simulated for two cases: (a) Compensating for the acoustic propagation delay and (b) Not compensating for acoustic propagation delay. Figures 4 and 5 compare the proposed particles and the final particles for the two cases.



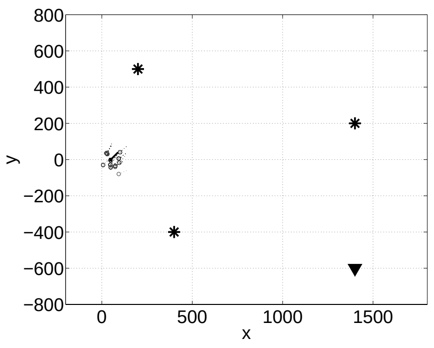
(a) With delay compensation.



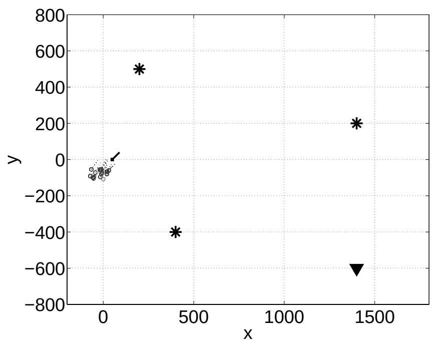
(b) Without delay compensation.

Fig. 4. Proposed particles with and without delay compensation.

It is seen in Figure 4 that when there is no compensation for acoustic propagation delay, the particles proposed by the acoustic nodes represent the target's state at a previous time. Hence,



(a) With delay compensation.

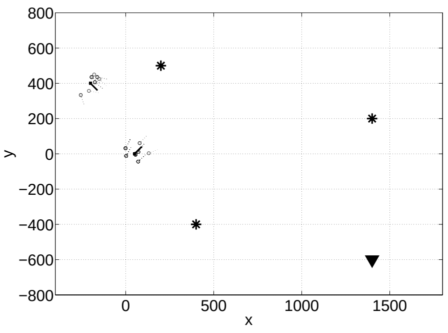


(b) Without delay compensation.

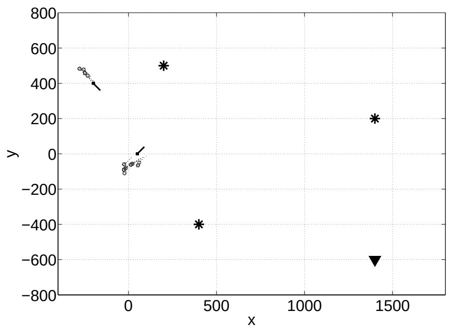
Fig. 5. Final set of particles using our initialization algorithm.

the four sensor estimates do not agree in the correct x-y location. This problem is eliminated when the particles proposed by the acoustic nodes are propagated through the forward model to compensate for the propagation delay. In this case, the sensor estimates overlap in the region of the target's true state.

Figure 5 shows the particles that survive the final resampling step. These particles can be used to initialize a joint tracking algorithm. For case (a), the resulting particles have mean $\mathbf{x}_C = [57.1,\ -4.5,\ 47.3,\ 56.8]^T$. For case (b), the resulting particles have mean $\mathbf{x}_{NC} = [-27.6,\ -53.5,\ 40.9,\ 57.2]^T$. When compared to the true target state $\mathbf{x}_1$, it is clear that compensating for the acoustic propagation delay is essential for accurate initialization.



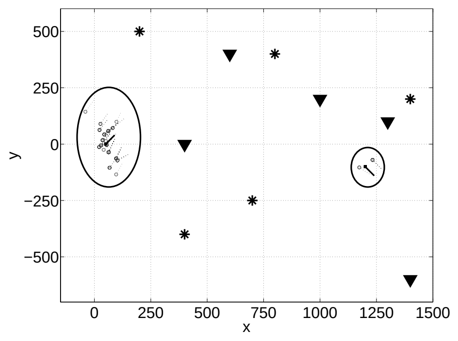
(a) With delay compensation.



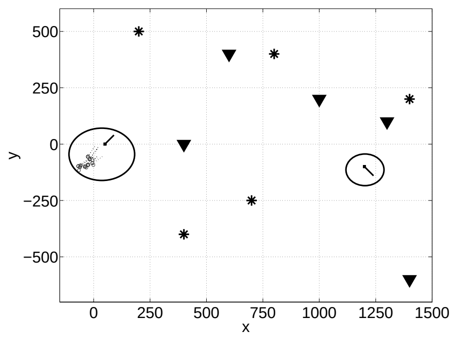
(b) Without delay compensation.

Fig. 6. Final set of particles for a multi-target example, using our initialization strategy.

The algorithm was simulated for the multi-target case when a second target is born at $\mathbf{x}_2 = [-200,\ 400,\ 50,\ -50]^T$. The same two cases discussed earlier were simulated. Figure 6 compares the particles that survive the final resampling step. It can be seen in Fig. 6(b) that the final set of particles lag behind the true targets' states. However, in Fig. 6(a), the particles are distributed in the vicinity of the true targets' states. Once again it is clear that compensating for the acoustic propagation delay is essential for accurate initialization.



(a) With delay compensation.



(b) Without delay compensation.

Fig. 7. Final set of particles for a large network.

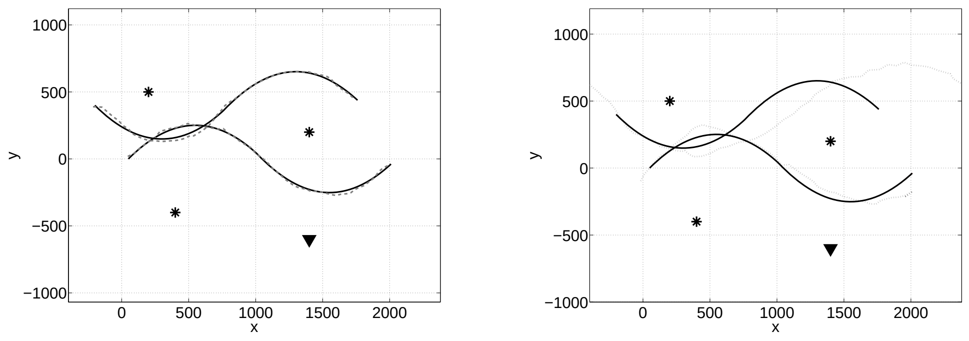
Figure 7 shows simulation results in a larger network with 10 sensors. Two targets are born with initial states $\mathbf{x}_1 = [50,\ 0,\ 40,\ 40]^T$ and $\mathbf{x}_2 = [1200,\ -100,\ 40,\ -40]^T$. There are 5 radar nodes and 5 acoustic arrays at the locations shown in Figure 7. The acoustic arrays at (200 m, 500 m) and (400 m, -400 m) and the radar node at (400 m, 0 m) only see target 1. The radar nodes at (1400 m, -600 m) and (1300 m, 100 m) and the acoustic array at (1400 m, 200 m) only see target 2. The remaining sensors see both targets. Fig. 7(a) shows simulation results with delay compensation and Fig. 7(b) shows results without delay compensation. As expected, the results with delay compensation show correct initialization for both targets. The results in the uncompensated case not only lag behind the true targets' states, but also completely miss one of the targets. This behavior is persistent after repeating the simulation multiple times.

The final set of particles from Figure 6 were used to initialize the multi-target DJT described in Section VII. The targets' initial states and the sensor positions are the same as those described in the multi-target initialization example given earlier in this section. The ideas developed in Section VI are extended to the multi-target tracking case to compensate for the acoustic signal propagation delays. Fig. 8(a) shows a tracking example when (31) is used as the likelihood function. For comparison, Fig. 8(b) shows the same tracking example when acoustic signal propagation delay is not compensated for. In this case, (27) is used as the likelihood function. In both plots, the solid curves represent the true target tracks while the dotted curves represent the estimated tracks.

Fig. 8(b) shows one of the best results acquired without compensating for signal propagation delay. In most simulations, if the delay was not compensated for, the tracker would lose either one or both targets completely. Fig. 8(a) shows a typical tracking result when likelihood function (31) is used to compensate for the propagation delay. It can be seen clearly that in large networks with fast moving targets, signal propagation delay compensation is essential for accurate tracking.

IX. Conclusions and Future Work

A method for generating the initial probability distribution is proposed for multiple targets in a decentralized multi-modal smart sensor network. Our method takes into account missing data, clutter, processing delays, communication delays and signal propagation delays. Compensation is achieved by using the forward model of the targets' state evolution. A Monte-Carlo method



(a) With delay compensation. (b) Without delay compensation.

Fig. 8. Tracking example.

is used to sequentially sample the state space to generate particles and a robust weighting function is used to represent the degree of belief in each particle. This weighting function can accommodate multiple targets, clutter, missing data and delays. The final targets' state distribution is represented by a 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 decentralized tracking algorithms.

The ideas developed in this paper are extended to a decentralized multi-modal multi-target tracking scenario. Using the likelihood function developed in the initialization algorithm, a standard decentralized tracker has the ability to compensate for delayed estimates from a subset of its sensors.

For this paper, the prior targets' state distribution was assumed to be uninformative and chosen to be uniform. Future work will also consider the case of informative priors to generate distributions reflecting prior knowledge of the true targets' states.

APPENDIX I
INITIALIZATION ALGORITHM

- **Variables:**
  D = Number of particles used for initialization.
  $S(i)$ = Sensor i, where $i = 1, ..., M$
  $K(i)$ = Target i, where $i = 1, ..., K$
  w_{num} = Numerator of weights.
  w_{den} = Denominator of weights.
  w = particle weights.
  $x_t^{(i)}$ = particle i at time t.
- **STEP 1: Sequentially Sampling the Proposal Function**
  $\widetilde{w}$ = 0
  If $S(1)$ has a detection,
  – Sample D particles in the X-Y space based on the posterior distribution at $S(1)$
  – Each particle will have equal weight
  – $\widetilde{w} = \widetilde{w} + 1$

```
    Else
      – set all particles equal to 0
    Send particles and $\widetilde{w}$ to $S(2)$
    For $i = 2, ..., M$
      – Current sensor is $S(i)$
      – Accept $D$ particles and $\widetilde{w}$ from $S(i-1)$
      – Give each received particle a weight of $\widetilde{w}$
      – If $S(i)$ has a detection
          ∗ Sample $D$ new particles based on the posterior distribution at $S(i)$
          ∗ Each new particle will have equal weight
              · Give each new particle a weight of 1
          ∗ From the $2D$ particles, obtain $D$ particles by using a weighted sampling with
            replacement.
          ∗ Each particle will now have equal weight
          ∗ $\widetilde{w} = \widetilde{w} + 1$
      – Send particles and $\widetilde{w}$ to $S(i+1)$
• **STEP 2: Weighting the Particles and Back Propagating Final Particles**
    For $i = 0, ..., M-1$
      – Current sensor is $S(M-i)$
      – If $i > 0$
          ∗ Accept particles, $w_{\text{num}}$ and $w_{\text{den}}$ from $S(M-i+1)$
      – Else
          ∗ $w_{\text{num}} = 1$
          ∗ $w_{\text{den}} = 0$
      – For $i = 1, ..., D$
          ∗ $w_{\text{num}}^{(i)} = w_{\text{num}}^{(i)} \cdot p(\mathbf{z}_{t,M-i}|x_t^{(i)})$
          ∗ $w_{\text{den}}^{(i)} = w_{\text{den}}^{(i)} + \frac{p(\mathbf{z}_{t,M-i}|x_t^{(i)})}{p(\mathbf{z}_{t,M-i})}$
      – Send particles, $w_{\text{num}}$ and $w_{\text{den}}$ to $S(M-i-1)$
• **STEP 3: Propagating Final Weights**
    Current sensor is $S(1)$
    For $i = 1, ..., D$
      – $w^{(i)} = \frac{w_{\text{num}}^{(i)}}{w_{\text{den}}^{(i)}}$
    Send $w$ to $S(2)$
    For $i = 2, ..., M$
      – Accept $w$ from $S(i-1)$
      – Send $w$ to $S(i+1)$
```

REFERENCES

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

[2] G.J. Pottie and W.J. Kaiser, “Wireless integrated network sensors,” *Communications of the ACM*, vol. 43, pp. 51–58, May 2000.

[3] Y. Wong, J. Wu, L. Ngoh, and W. Wong, "Collaborative data fusion tracking in sensor networks using monte carlo methods," in *Proceedings. 29th Annual IEEE International Conference on Local Computer Networks*, 2004.

[4] M. Liggins II, C. Chong, I. Kadar, M. Alford, V. Vannicola, and S. Thomopoulos, "Distributed fusion architectures and algorithms for target tracking," in *Proceedings of the IEEE*, 1997.

[5] P. Storms, J van Veelen, and E. Boasson, "A process distribution approach for multisensor data fusion systems based on geographical dataspace partitioning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, pp. 14–23, Jan. 2005.

[6] S. Balasubramanian, I. Elangovan, S. Jayaweera, and K. Namuduri, "Distributed and collaborative tracking for energy-constrained ad-hoc wireless sensor networks," *IEEE Wireless Communications and Networking Conference*, vol. 3, pp. 1732–7, 2004.

[7] J. Liu, M. Chu, J. Liu, J. Reich, and F. Zhao, "Distributed state representation for tracking problems in sensor networks," *Third International Symposium on Information Processing in Sensor Networks*, pp. 234–42, 2004.

[8] I. Leichter, M. Lindenbaum, and E. Rivlin, "A probabilistic framework for combining tracking algorithms," in *CVPR 2004*, WDC, June 27–July 2 2004.

[9] V. Cevher and J. H. McClellan, "General direction-of-arrival tracking with acoustic nodes," *IEEE Trans. on Signal Processing*, vol. 53, pp. 1–12, Jan. 2005.

[10] M. Orton and W. Fitzgerald, "A Bayesian approach to tracking multiple targets using sensor arrays and particle filters," *IEEE Trans. on Signal Processing*, vol. 50, no. 2, pp. 216–223, February 2002.

[11] R.R. Allen and S.S. Blackman, "Implementation of an angle-only tracking filter," in *SPIE Proc.*, 1991, vol. 1481, pp. 292–303.

[12] A. Farina, "Target tracking with bearings-only measurements," *Elsevier Signal Processing*, vol. 78, pp. 61–78, 1999.

[13] Y. Zhou, P.C. Yip, and H. Leung, "Tracking the direction-of-arrival of multiple moving targets by passive arrays: Algorithm," *IEEE Trans. on Signal Processing*, vol. 47, no. 10, pp. 2655–2666, October 1999.

[14] J. Sanchez-Araujo and S. Marcos, "An efficient PASTd-algorithm implementation for multiple direction of arrival tracking," *IEEE Trans. on Signal Processing*, vol. 47, pp. 2321–2324, August 1999.

[15] V.J. Aidala, "Kalman filter behavior in bearings-only tracking applications," *IEEE Trans. on Aerospace and Electronic Systems*, vol. AES-15, pp. 29–39, January 1979.

[16] S. Hong, R. Evans, and H. Shin, "Optimization of waveform and detection threshold for range and range-rate tracking in clutter," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, pp. 17–33, 2005.

[17] E. Hughes and M. Lewis, "Intelligent agents for radar systems," *Electronics Systems and Software*, vol. 3, pp. 39–43, Feb.-March 2005.

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

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

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

[21] M. Isard and A. Blake, "Condensation – conditional density propagation for visual tracking," *International Journal of Computer Vision*, vol. 29, pp. 5–28, 1998.

[22] M.J. Coates, "Distributed particle filtering for sensor networks," *International Symposium on Information Processing in Sensor Networks*, 2004.