DISTRIBUTED INITIALIZATION OF SENSOR NETWORKS WITH COMMUNICATION AND COMPUTATION TREES

Milind Borkar and James H. McClellan

Center for Signal and Image Processing
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0250

ABSTRACT

When compared to the tracking problem in which prior knowledge is available, generating the initial distribution for the state vector of a phenomenon of interest, with no prior knowledge of the desired state, is a challenging problem. In this paper, the authors develop a fully distributed initialization algorithm that fuses data in heterogeneous sensor networks using communication trees. Monte Carlo methods are used to fuse the collected data and to represent the desired state vector distribution. The presented algorithm utilizes an importance function that is additive in the local node posterior distributions, providing a robust alternative to belief propagation methods in which particles are generated according to the product of local node posteriors.

***Index Terms*—** Multisensor systems, data fusion, initialization, distributed processing, Monte Carlo methods

1. INTRODUCTION

The algorithm presented in this paper focuses on distributed data fusion in arbitrary sensor networks, possibly consisting of heterogeneous sensor nodes. These nodes can not only sense the environment, but can also (i) process sensed data to produce local estimates, referred to as *organic* state estimates, and (ii) communicate with neighboring nodes. To ensure scalability, the communication bandwidth should be fixed. The goal is to generate the initial probability density function (pdf) of the state vector of a phenomenon of interest, called the target state. This pdf can be used as the required prior knowledge to initialize arbitrary tracking algorithms. Since an analytical solution is complicated, the initialization algorithm presented in this paper uses a sequential implementation of importance sampling [1], a powerful Monte Carlo method.

The presented algorithm has general applicability in arbitrary sensor networks because it only depends on the ability of the network to organize its communication in a tree structure—any node could play the role of the root node in the tree. The root node would be responsible for launching the initialization algorithm. It would process the data it receives from the environment and produce an estimate of the pdf of the target's state. This estimate is generated in a discrete format using particles, representing hypothesized target states, and their associated importance weights, representing the degree of belief in these particles. These particles and importance weights are then sent to one or more neighboring nodes which are the children of the root node. Each neighboring (child) node generates its own intrinsic set of particles and importance weights based on its organic state estimates. Then a combining step is needed to merge the intrinsic particles and importance weights with the received set of particles and importance weights to produce an improved representation of the target's state distribution. This process is obviously recursive so it can be continued as particles propagate throughout the network from the root node to its descendants (children, grandchildren, and so on). Once the leaf nodes of the tree are reached, it is necessary to propagate the pdf's back up to the root, so that one node will have a pdf that incorporates the measurements from all nodes. In the process of moving the pdf's back up the tree it is necessary to merge pdf's from different branches while not weighting any measurement more than once. The last step in the initialization is a global broadcast of the final pdf from the root to all the other nodes.

The initialization algorithm presented in this work uses an equally weighted mixture of local node posterior distributions as the sequentially generated importance function. This provides an attractive alternative to the commonly used nonparametric implementations of belief propagation [2, 3] in which particles are sequentially generated according to the product of local node posteriors. While the belief propagation methods provide high resolution in regions of the target state space where all sensor nodes are in agreement, our initialization algorithm saves local knowledge from being lost during the sequential updates, which could prove extremely useful in real world scenarios with missed detections and false alarms.

In the rest of the paper, Section 2 briefly describes pre-

Sponsored by the U. S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-02-0008.

vious work by the authors addressing distributed initialization of heterogeneous sensor networks with a chain-structured communication topology. In Section 3, the initialization algorithm is generalized to more realistic communication topologies for the network. In Section 4, the new algorithm is simulated in a sensor network consisting of bearing-only nodes and range-only nodes. Section 5 summarizes the results.

2. CHAIN STRUCTURED ALGORITHM

Consider the basic case in which the sensors in the network communicate with each other using a fixed one-hop chain. In previous work [4] the authors developed an algorithm to effectively fuse data collected at the various sensor nodes in a fully distributed manner. The algorithm uses importance sampling to sequentially generate a set of particles and importance weights that represent the combined knowledge of all the nodes in the network. If $\mathbf{s}_t$ represents the target state vector to be approximated at time t, $\mathbf{z}_t$ represents the joint set of organic state estimates from all the M sensors in the network, and $\mathbf{z}_{m,t}$ represents the vector of organic state estimates at the m^{th} sensor, then the importance function is given by a sum

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

Assuming conditional independence of estimates available at the various nodes given the true target state, the importance weights $\{w_t^{(i)}\}_{i=1}^{D}$ assigned to particles $\{\mathbf{s}_t^{(i)}\}_{i=1}^{D}$ are

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

It can be shown [4] that these particles and importance weights together represent the desired posterior distribution

$$p(\mathbf{s}_t|\mathbf{z}_t) \propto \prod_{m=1}^{M} p(\mathbf{s}_t|\mathbf{z}_{m,t}). \tag{3}$$

Internode messages consist of sets of D particles and importance weights.

3. TREE STRUCTURED ALGORITHM

The goal of this paper is to generalize the initialization algorithm developed in [4] to sensor networks using communication trees for message passing. A root node launches the initialization algorithm and propagates messages downward in the root-to-leaf direction. During this downward pass, data from various nodes is aggregated, and the internode messages represent the combined knowledge for the branches of the tree that have already been traversed. Once the messages reach the leaf nodes, they are propagated upwards back to the root. During this upward pass, data across branches is aggregated, and the internode messages represent the combined knowledge for the different subtrees from multiple children. The final set of messages that reach the root node represent the joint knowledge of the entire tree, and is disseminated through the network. The fusion algorithms for the downward pass and the upward pass are developed in the following subsections.

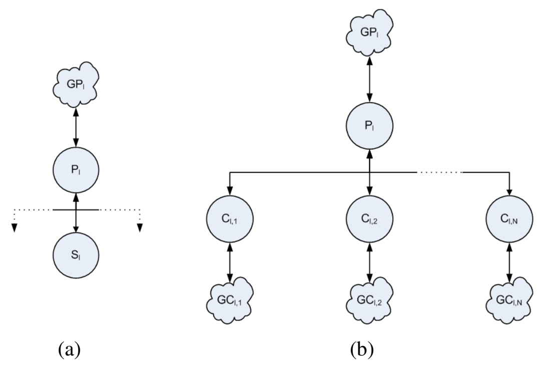

Fig. 1. Subtrees for algorithm development. Circles represent single nodes and clouds represent sets of nodes: (a) Downward pass (b) Upward pass.

3.1. Downward Pass

Consider a leaf node S$_l$ with parent node P$_l$ and a set of ancestor nodes GP$_l$. The root node is contained within GP$_l$. A message launched by the root node propagates downward through the network, incorporating data from nodes that it traverses. The particular message received at S$_l$ has followed a fixed one-hop communication chain starting from the root node as given in Fig. 1(a). A distributed fusion algorithm for such a chain has been developed in [4]. After fusing the received message with local organic state estimates, the particles and importance weights available at S$_l$ are given by (4) and (5), where M_l represents the number of nodes traversed by the message.

$$\mathbf{s}_{l,t}^{(i)} \sim \frac{1}{M_l}\sum_{m\in\{\mathrm{S}_l,\mathrm{P}_l,\mathrm{GP}_l\}} p(\mathbf{s}_t|\mathbf{z}_{m,t}), \tag{4}$$

$$w_{l,t}^{(i)} \propto \frac{\prod_{m\in\{\mathrm{S}_l,\mathrm{P}_l,\mathrm{GP}_l\}} p(\mathbf{s}_{l,t}^{(i)}|\mathbf{z}_{m,t})}{\sum_{m\in\{\mathrm{S}_l,\mathrm{P}_l,\mathrm{GP}_l\}} p(\mathbf{s}_{l,t}^{(i)}|\mathbf{z}_{m,t})}. \tag{5}$$

3.2. Upward Pass

During the upward pass shown in Fig. 1(b), a parent node P$_l$ receives messages from multiple child nodes C$_{l,n}$, with

$n = 1, \ldots, N$. The fusion algorithm developed below is the general method to fuse the messages received by any arbitrary node in the network, P_l, whose ancestor nodes are the set GP_l. Without loss of generality, one can assume that a message sent by the child $\mathrm{C}_{l,n}$ to P_l represents the joint knowledge of all subtrees below it, defined by the set of all successor nodes $\mathrm{GC}_{l,n}$, and common ancestors. In the following equations, M_* represents the total number of nodes that have been traversed by the current message and is used simply for normalization. Thus, the message received by P_l from $\mathrm{C}_{l,n}$ consists of particles and importance weights given by

$$\mathbf{s}^{(i)}_{\mathrm{C}_{l,n},t} \sim \frac{1}{M_{l,n}} \sum_{m \in \{\mathrm{GP}_l, \mathrm{P}_l, \mathrm{C}_{l,n}, \mathrm{GC}_{l,n}\}} p(\mathbf{s}_t|\mathbf{z}_{m,t}), \tag{6}$$

$$w^{(i)}_{\mathrm{C}_{l,n},t} \propto \frac{\prod_{m \in \{\mathrm{GP}_l, \mathrm{P}_l, \mathrm{C}_{l,n}, \mathrm{GC}_{l,n}\}} p(\mathbf{s}^{(i)}_{\mathrm{C}_{l,n},t}|\mathbf{z}_{m,t})}{\sum_{m \in \{\mathrm{GP}_l, \mathrm{P}_l, \mathrm{C}_{l,n}, \mathrm{GC}_{l,n}\}} p(\mathbf{s}^{(i)}_{\mathrm{C}_{l,n},t}|\mathbf{z}_{m,t})}. \tag{7}$$

Particles and importance weights from the first downward pass, representing common knowledge of the ancestors, are still stored at P_l. These are given by

$$\widehat{\mathbf{s}}^{(i)}_{\mathrm{P}_l,t} \sim \frac{1}{M_{\mathrm{P}_l}} \sum_{m \in \{\mathrm{GP}_l, \mathrm{P}_l\}} p(\mathbf{s}_t|\mathbf{z}_{m,t}), \tag{8}$$

$$\widehat{w}^{(i)}_{\mathrm{P}_l,t} \propto \frac{\prod_{m \in \{\mathrm{GP}_l, \mathrm{P}_l\}} p(\widehat{\mathbf{s}}^{(i)}_{\mathrm{P}_l,t}|\mathbf{z}_{m,t})}{\sum_{m \in \{\mathrm{GP}_l, \mathrm{P}_l\}} p(\widehat{\mathbf{s}}^{(i)}_{\mathrm{P}_l,t}|\mathbf{z}_{m,t})}. \tag{9}$$

After fusion is complete, the final set of particles and importance weights should obey

$$\mathbf{s}^{(i)}_{\mathrm{P}_l,t} \sim \frac{1}{M_l} \sum_{m \in \{\mathrm{GP}_l, \mathrm{P}_l, \mathrm{C}_{l,1,\ldots,N}, \mathrm{GC}_{l,1,\ldots,N}\}} p(\mathbf{s}_t|\mathbf{z}_{m,t}), \tag{10}$$

$$w^{(i)}_{\mathrm{P}_l,t} \propto \frac{\prod_{m \in \{\mathrm{GP}_l, \mathrm{P}_l, \mathrm{C}_{l,1,\ldots,N}, \mathrm{GC}_{l,1,\ldots,N}\}} p(\mathbf{s}^{(i)}_{\mathrm{P}_l,t}|\mathbf{z}_{m,t})}{\sum_{m \in \{\mathrm{GP}_l, \mathrm{P}_l, \mathrm{C}_{l,1,\ldots,N}, \mathrm{GC}_{l,1,\ldots,N}\}} p(\mathbf{s}^{(i)}_{\mathrm{P}_l,t}|\mathbf{z}_{m,t})}. \tag{11}$$

One can define a set of scaled weights as

$$\widetilde{w}^{(i)}_t = \prod_{m \in \{\mathrm{GP}_l, \mathrm{P}_l, \mathrm{C}_{l,1,\ldots,N}, \mathrm{GC}_{l,1,\ldots,N}\}} p(\mathbf{s}^{(i)}_t|\mathbf{z}_{m,t}). \tag{12}$$

After some algebraic manipulation and assuming conditional independence of estimates at the various nodes given the true target state, the scaled weights can be written as

$$\widetilde{w}^{(i)}_t = \frac{\prod_{n=1}^{N} \left(\prod_{m \in \{\mathrm{GP}_l, \mathrm{P}_l, \mathrm{C}_{l,n}, \mathrm{GC}_{l,n}\}} p(\mathbf{s}^{(i)}_t|\mathbf{z}_{m,t}) \right)}{\left(\prod_{m \in \{\mathrm{GP}_l, \mathrm{P}_l\}} p(\mathbf{s}^{(i)}_t|\mathbf{z}_{m,t}) \right)^{N-1}}. \tag{13}$$

Since particles and importance weights representing the numerator and denominator are given in (6), (7), (8) and (9), scaled weights can be evaluated using kernel density estimation [5] as

$$\widetilde{w}^{(i)}_t = \frac{\prod_{n=1}^{N} \left(\sum_{j=1}^{D} w^{(j)}_{\mathrm{C}_{l,n},t} W\left(\mathbf{s}^{(i)}_t - \mathbf{s}^{(j)}_{\mathrm{C}_{l,n},t}\right) \right)}{\left(\sum_{j=1}^{D} \widehat{w}^{(j)}_{\mathrm{P}_l,t} W\left(\mathbf{s}^{(i)}_t - \widehat{\mathbf{s}}^{(j)}_{\mathrm{P}_l,t}\right) \right)^{N-1}}, \tag{14}$$

where $W(\cdot)$ is an appropriate stochastic kernel. Such scaled weights are evaluated for all received particles and stored as $\{\{\widetilde{w}^{(i)}_{\mathrm{C}_{l,n},t}\}^{D}_{i=1}\}^{N}_{n=1}$.

Next, particles representing (10) must be generated from the ND received particles $\{\{\mathbf{s}^{(i)}_{\mathrm{C}_{l,n},t}\}^{D}_{i=1}\}^{N}_{n=1}$. This can be accomplished using a weighted resampling operation. Each set of received particles can be divided into two sets, and the resampling weights $\breve{w}$, not to be confused with the importance weights, can be assigned accordingly. Let $M_{l,n}$ denote the number of nodes represented in the message received from $\mathrm{C}_{l,n}$. The first set represents particles sampled from the child and successor posteriors only. This set of particles from $\mathrm{C}_{l,n}$ is given by

$$\{\mathbf{s}^{(j)}_{\mathrm{C}_{l,n},\mathrm{C},t}\} = \{\mathbf{s}^{(i)}_{\mathrm{C}_{l,n},t}\}^{D}_{i=1} \bigcap \overline{\{\widehat{\mathbf{s}}^{(i)}_{\mathrm{P}_l,t}\}^{D}_{i=1}}, \tag{15}$$

and assigned resampling weights

$$\{\breve{w}^{(j)}_{\mathrm{C}_{l,n},\mathrm{C},t}\} = M_{l,n}. \tag{16}$$

The second set represents particles sampled from the parent and ancestor posteriors only. This set of particles from $\mathrm{C}_{l,n}$ is given by

$$\{\mathbf{s}^{(j)}_{\mathrm{C}_{l,n},\mathrm{P},t}\} = \{\mathbf{s}^{(i)}_{\mathrm{C}_{l,n},t}\}^{D}_{i=1} \bigcap \{\widehat{\mathbf{s}}^{(i)}_{\mathrm{P}_l,t}\}^{D}_{i=1}, \tag{17}$$

and assigned resampling weights

$$\{\breve{w}^{(j)}_{\mathrm{C}_{l,n},\mathrm{P},t}\} = \frac{M_{l,n}}{N}. \tag{18}$$

A set of D particles can be generated by weighted sampling with replacement from the sets of particles $\{\{\mathbf{s}^{(i)}_{\mathrm{C}_{l,n},t}\}^{D}_{i=1}\}^{N}_{n=1}$ using resampling weights generated using (16) and (18). The surviving particles and the associated scaled weights are labeled and saved as $\{\mathbf{s}^{(i)}_{\mathrm{P}_l,t}, \widetilde{w}^{(i)}_{\mathrm{P}_l,t}\}^{D}_{i=1}$.

The scaled weights $\{\widetilde{w}^{(i)}_{\mathrm{P}_l,t}\}^{D}_{i=1}$ can now be modified to determine the importance weights $\{w^{(i)}_{\mathrm{P}_l,t}\}^{D}_{i=1}$. From (11) and (12), the importance weights can be defined as

$$w^{(i)}_{\mathrm{P}_l,t} \propto \frac{\widetilde{w}^{(i)}_{\mathrm{P}_l,t}}{\sum_{m \in \{\mathrm{GP}_l, \mathrm{P}_l, \mathrm{C}_{l,1,\ldots,N}, \mathrm{GC}_{l,1,\ldots,N}\}} p(\mathbf{s}^{(i)}_{\mathrm{P}_l,t}|\mathbf{z}_{m,t})}, \tag{19}$$

and can be evaluated using kernel density estimation as

$$w_{\mathrm{P}_l,t}^{(i)} \propto \frac{\widetilde{w}_{\mathrm{P}_l,t}^{(i)}}{\sum_{j=1}^{D} W\left(\mathbf{s}_{\mathrm{P}_l,t}^{(i)} - \mathbf{s}_{\mathrm{P}_l,t}^{(j)}\right)}. \tag{20}$$

The final particles and importance weights $\{\mathbf{s}_{\mathrm{P}_l,t}^{(i)}, w_{\mathrm{P}_l,t}^{(i)}\}_{i=1}^{D}$ at P_l represent the desired joint distribution

$$p(\mathbf{s}_t|\mathbf{z}_{\mu,t}),\ \mu = \{m|m \in \{\mathrm{GP}_l, \mathrm{P}_l, \mathrm{C}_{l,1,\ldots,N}, \mathrm{GC}_{l,1,\ldots,N}\}\}, \tag{21}$$

and are propagated to the parent node for P_l. This fusion procedure is repeated at each node during the upward pass until messages arrive at the root node. After fusing incoming messages at the root node, the final particles and importance weights $\{\mathbf{s}_t^{(i)}, w_t^{(i)}\}_{i=1}^{D}$ represent the joint distribution for the entire network, and need only be broadcast to complete the distributed initialization.

4. SIMULATION

The generalized initialization algorithm is simulated in a network consisting of bearing-only nodes and range-only nodes as shown in Fig. 2(a). None of these nodes are capable of locally observing the states, $\mathbf{s}_t = [x_t,\ y_t]^{\mathrm{T}}$, representing the initial positions, in meters, of two targets, T1 and T2, that are seen simultaneously by the sensor network. A total of 7 nodes exist, S1 through S7. S1 is the root node that launches the initialization algorithm. The paths taken by various messages form a tree. All nodes except S2 and S3 see both targets. S2 only sees T1 and S3 only sees T2. The algorithm developed in Section 3 is used to fuse noisy local estimates in a distributed manner, with minor modifications made to account for missed detections. Explicit data association is not required.

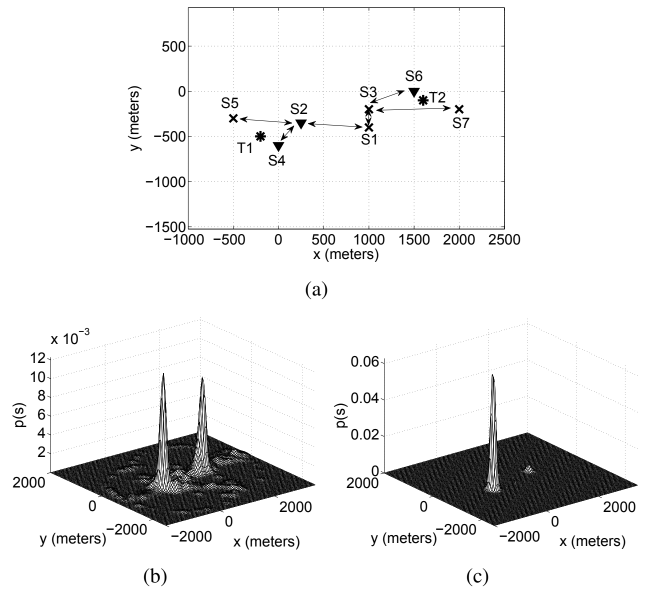

Fig. 2. (a) Sensor network setup. $\times$ represent bearing nodes, $\bigtriangledown$ represent range nodes, $*$ represent targets, and $\leftrightarrow$ represent communication paths. (b) Joint distribution generated using our algorithm. (b) Joint distribution generated using a nonparametric implementation of belief propagation.

The joint distribution for the entire network generated using the particles and importance weights at the end of the upward pass is plotted in Fig. 2(b). It can be seen that two distinct peaks appear at the true target locations. This demonstrates that the initialization algorithm developed in this paper is effective in fusing data across the network. A nonparametric implementation of belief propagation fails to generate peaks at both target states, as shown in the pdf given in Fig. 2(c).

5. SUMMARY

The authors have developed a distributed initialization algorithm for sensor networks using communication trees. The presented algorithm uses Monte Carlo methods to fuse data collected at the various nodes in the network and to represent the initial target state vector distribution. Simulation results for sensor networks consisting of bearing nodes and range nodes show good initialization performance, even in the case where targets are occluded. A nonparametric implementation of belief propagation fails under similar conditions.

6. REFERENCES

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

[2] 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*, 2003.

[3] 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*, 2003.

[4] M. Borkar, V. Cevher, and J. H. McClellan, "Low computation and low latency algorithms for distributed sensor network initialization," *Signal, Image and Video Processing Journal (Springer)*, vol. 1, no. 2, pp. 133–148, June 2007.

[5] E. Parzen, "On estimation of a probability density function and mode," *Annals of Mathematical Statistics*, vol. 33, pp. 1065–1076, 1962.