**Preprints of the 19th World Congress**
**The International Federation of Automatic Control**
**Cape Town, South Africa. August 24-29, 2014**

# Distributed Monte Carlo Information Fusion and Distributed Particle Filtering

**Isaac L. Manuel and Adrian N. Bishop**

*Australian National University and NICTA, Canberra Research Lab*
*(e-mail: isaac@isaacmanuel.com and adrian.bishop@anu.edu.au)*

**Abstract:** We present a Monte Carlo solution to the distributed data fusion problem and apply it to distributed particle filtering. The consensus-based fusion algorithm is iterative and it involves the exchange and fusion of empirical posterior densities between neighbouring agents. As the fusion method is Monte Carlo based it is naturally applicable to distributed particle filtering. Furthermore, the fusion method is applicable to a large class of networks including networks with cycles and dynamic topologies. We demonstrate both distributed fusion and distributed particle filtering by simulating the algorithms on randomly generated graphs.

## 1. INTRODUCTION

As modern sensor systems move towards distributed architectures it is important to consider the design of practical algorithms for data estimation and fusion. Such algorithms should be robust and promote network scalability.

In this paper we develop a distributed method for data fusion over networks and apply the algorithm to distributed particle filtering. Our approach to this problem is consensus-based and involves the local exchange of posterior densities. The posteriors are exchanged via sets of unweighted particles. The algorithm converges to the optimal (centralised) solution if the likelihood functions are conditionally independent. If they are dependent then our algorithm can be modified to return a posterior that is guaranteed to be consistent [Bailey et al. (2012)].

There are many attractive aspects to our approach. (i) the method is distributed: the agents share and compute with local knowledge. (ii) the fusion scheme is robust to networks containing cycles and networks with time varying topologies. (iii) the algorithm has a guaranteed speed of convergence. (iv) the resulting posterior distribution is guaranteed to be consistent as long as the dependence between the agents is understood.

The algorithm we present is falls under the category of consensus-based distributed particle filtering. An alternative to consensus is message passing (or routing) based information sharing. The following is a short review of some of the literature in this space.

The distributed fusion method of Hlinka et al. (2012) (see also Hlinka et al. (2011) and Hlinka et al. (2010)) relies on the individual agent's likelihood functions belonging to the exponential family of distributions. They apply a consensus scheme to approximations of the exponential family and arrive at a joint likelihood function. The consensus method of Gu (2007) involves sharing the parameters of Gaussian mixture models. The approaches of Sheng et al. (2005) and Hlinka et al. (2009) also transmit the parameters of a Gaussian mixture model. However their fusion schemes are not based upon consensus, but rather message passing. Gu et al. (2008), another consensus-based fusion scheme, simplifies the parameter exchange by transmitting only the mean and covariance. Consequently this supports Gaussian dynamic systems, however they propose its use for systems with higher order moments by applying the unscented transform. Mohammadi and Asif (2011a) also take advantage of the unscented transform and propose its use to incorporate distributed Kalman filters into networks of distributed particle filters.

Mohammadi and Asif (2011b) propose another consensus scheme where each node runs two particle filters: a local particle filter, and a fusion filter, where the fusion filter computes the global distribution. Üstebay et al. (2011) use a selective gossip procedure to arrive at a consensus about the likelihoods of particles. The selective gossip procedure shares particles based upon weights. This focuses communication on the particles that contain the most information. Oreshkin and Coates (2010) also use a gossip consensus approach to share Gaussian approximations of the posterior. Recent work by Lee and West (2013) proposes a Markov Chain Distributed Particle Filter (see additionally Lee and West (2009) and Lee and West (2010)) in which neighbours exchange particles and weights using a Markov chain random walk.

Further approaches to distributed particle filtering are outlined in Bashi et al. (2003), Sheng and Hu (2005), Bolić et al. (2005), Mohammadi and Asif (2009), Farahmand et al. (2011), and Savic et al. (2012). In addition, the review by Hlinka et al. (2013) is an excellent resource for comparing methods of distributed particle filtering.

This paper is organised as follows. Section 2 briefly discuses the distributed data fusion problem, then details the proposed Monte Carlo based fusion algorithm. The algorithm is supported by two simulations. Section 3 considers the application of the previously derived distributed Monte

[1] The authors were supported by NICTA and A.N. Bishop is also supported by the Australian Research Council via a Discovery Early Career Researcher Award (DE-120102873) and by the US Air Force via the Asian Office of Aerospace Research and Development (AOARD) and contract USAF-AOARD-144042.

Carlo fusion algorithm to the problem of distributed particle filtering. In Section 4 we offer conclusions.

## 2. DISTRIBUTED MONTE CARLO DATA FUSION

Consider a group of agents indexed in $\mathcal{V} = \{1, \ldots, n\}$ and a set of possible time-varying undirected links $\mathcal{E}(k) \subset \mathcal{V} \times \mathcal{V}$ defining a network graph $\mathcal{G}(\mathcal{V}, \mathcal{E}(k))$. The neighbor set at agent $i$ is denoted by $\mathcal{N}_i(k) = \{j \in \mathcal{V} : (i,j) \in \mathcal{E}(k)\}$ and $j \in \mathcal{N}_i(k) \Leftrightarrow i \in \mathcal{N}_j(k)$ for undirected topologies. We often drop the network's dependence on $k$ for brevity.

Each agent constructs an initial local posterior of the form

$$p(x|y_i) = \frac{g(y_i|x)p(x)}{\int g(y_i|x)p(x)dx} \propto g(y_i|x)p(x) \tag{1}$$

from given measurements $y_i \in \mathbb{R}^{m_{y_i}}$ and where $g(y_i|x)$ is the likelihood function at agent $i$ conditioned on some underlying event $x \in \mathbb{R}^{m_x}$. Here, $p(x)$ is the prior information common to all agents. The goal of distributed data fusion in this case is to compute

$$p(x|\{y_i\}_{i\in V}) \propto p(x) \prod_{i\in\mathcal{V}} g(y_i|x) \tag{2}$$

locally at each agent $i$ under the constraint that agent $i$ can only share $p(x|y_i)$ with its neighbours in $\mathcal{N}_i(t)$. In other words, each agent is constrained to computations involving local posteriors, $p(x|y_i)$ and $p(x|y_j)$ where $j \in \mathcal{N}_i$. Obviously, an iterative procedure is required to reach $p(x|\{y_i\}_{i\in V})$ at each agent in an incomplete network.

The following theorem is a slight modification of the main result in Olfati-Saber et al. (2006).

*Theorem 1.* Consider a network $\mathcal{G}(t)$ as described above where each agent exchanges $\pi_k^i$ with $\pi_{k=0}^i = p(x|y_i)$. Here $\pi_k^i$ and $\pi_k^j$ are *not conditionally independent*. Suppose $p(x) > 0$ and $g(y_i|x) > 0$ for all $i \in \mathcal{V}$. Then the following statements hold:

i The agents are capable of asymptotically reaching a consensus on $Q \propto p(x) \prod_{j\in V} g(y_j|x)^{1/n}$.

ii The consensus algorithm for agreement in the value $Q$ takes the form

$$\pi_k^i = (\pi_{k-1}^i)^{1-|\mathcal{N}_i|\gamma} \prod_{j\in\mathcal{N}_i} (\pi_{k-1}^j)^{\gamma}. \tag{3}$$

where $0 < \gamma < 1/\max(\{|\mathcal{N}_i| : i \in \mathcal{V}\})$ [2].

**Proof.** It is shown in Olfati-Saber et al. (2006) that $\pi_k^i \to \prod_{j\in\mathcal{V}} (\pi_0^j)^{1/n}$ as $k \to \infty$. In other words, $\log(\pi_k^i) \to (\sum_{j\in\mathcal{V}} \log(\pi_0^j))/n$ as $k \to \infty$.

Note then in this case $\pi_{k=0}^i = p(x|y_i) \propto g(y_i|x)p(x)$ and

$$\begin{aligned} \log(\pi_k^i) &\to \frac{\sum_{j\in\mathcal{V}} \log(\pi_0^j)}{n} \\ &\propto \frac{n\log(p(x))}{n} + \frac{\sum_{j\in V} \log(g(y_j|x))}{n} \end{aligned} \tag{4}$$

[2] The restriction that $\gamma$ is less than the inverse of the maximum degree in the network is sufficient for each agent to converge to $Q$. See Olfati-Saber et al. (2006) or Olfati-Saber and Murray (2004).

Taking the exponential gives

$$Q \propto p(x)g(y_i|x)^{1/n} \tag{5}$$

which completes the proof. □

Note that despite the fact each agent is sharing its local posterior the consensus-based distributed fusion algorithm just depicted converges to $p(x)\prod_{j\in V} g(y_j|x)^{1/n}$ and not $p(x)^n \prod_{j\in V} g(y_j|x)^{1/n}$ or $p(x)^{1/n}\prod_{j\in V} g(y_j|x)^{1/n}$. Thus it is not 'double counting' the common prior information $p(x)$ and nor is it conservative in this common prior information. The algorithm is actually conservative in $\prod_{j\in V} g(y_j|x)^{1/n}$ which may be important for consistency as discussed in Bailey et al. (2012). In particular, if $y_i$ and $y_j$ were correlated (i.e. $g(y_i|x)$ and $g(y_j|x)$ were not conditionally independent) and this dependence was not considered then over-confident results may be obtained if one simply computes $\prod_{j\in V} g(y_j|x)$. Similarly, if $p(x)$ was counted multiple times then over-confident results are clearly obtained. The proposed algorithm converges to the log-linear opinion pool [see Abbas (2009); Bailey et al. (2012)] on the likelihoods multiplied by the common prior which is guaranteed consistent. It converges to this value in a distributed manner which generalises Bailey et al. (2012).

*Corollary 2.* Suppose that $g(y_i|x)$ and $g(y_j|x)$ are conditionally independent for all $i, j \in \mathcal{V}$. Define $\pi_{k=0}^i \propto p(x)g(y_i|x)^n$. The proposed consensus-based distributed fusion algorithm converges to $Q' \propto p(x)\prod_{i\in V} g(y_i|x)$ as $k \to \infty$. This is exactly the optimal centralised Bayesian result given $p(x|y_i) \propto g(y_i|x)p(x)$ where $g(y_i|x)$ and $g(y_j|x)$ are conditionally independent and $p(x)$ is the common prior information among all agents.

The algorithm can be rewritten in the form

$$\pi_k^i = \pi_{k-1}^i \prod_{j\in\mathcal{N}_i} \left(\frac{\pi_{k-1}^j}{\pi_{k-1}^i}\right)^{\gamma} \tag{6}$$

which shows that when consensus is reached the iterations reduce to $\pi_k^i = \pi_{k-1}^i$.

This algorithm has many appealing points. Firstly, it is distributed: the agents only share and compute with local knowledge. Secondly, this distributed fusion algorithm can deal with cycles and time-varying topologies (where, for example, the network may be disconnected for many time steps). Thirdly, the algorithm has a guaranteed speed of convergence that is characterised by the network's algebraic connectivity; see Olfati-Saber et al. (2006). Finally, as noted above, the proposed algorithm converges to the log-linear opinion pool multiplied by the common prior information (which is a conservative, guaranteed consistent, version of the optimal Bayesian result regardless of the dependence between $g(y_i|x)$ and $g(y_j|x)$). Moreover, with a slight modification to the initially shared data we can also achieve distributed convergence to the true optimal Bayesian fusion result as noted in the corollary (and this result is obviously the most desired when it is known $g(y_i|x)$ and $g(y_j|x)$ are conditionally independent).

*2.1 A Monte Carlo Implementation of the Distributed Fusion Algorithm*

Given the desirable nature of the proposed data fusion algorithm it remains to establish a Monte Carlo version which allows one to begin with an empirical estimate of $\pi^i_{k=0} \propto p(x)g(y_i|x)^n$ or $\pi^i_{k=0} \propto p(x)g(y_i|x)$ given by

$$\overline{\pi}^i_{k=0}(x) = \frac{1}{N_s}\sum_{\ell=1}^{N_s}\delta(x - x_\ell) \tag{7}$$

where $x_\ell \in \mathbb{R}^{m_x}$ are a set of independent and identically distributed samples of $p(x)g(y_i|x)^n$ or $p(x)g(y_i|x)$ etc.

The main contribution of this section is a Monte Carlo version of the algorithm outlined in the previous subsection that allows one to begin with $\overline{\pi}^i_{k=0}$. This is desirable when dealing with complex estimation problems and/or when $\pi^i_{k=0}$ is sourced from alternative Monte Carlo estimations as in Doucet et al. (2001).

So given $\overline{\pi}^j_{k-1}$, $\forall j \in \mathcal{N}_i \cup \{i\}$ we want to compute $\overline{\pi}^i_k$ at each agent $i$ in the sense that $\overline{\pi}^i_k$ should be an empirical version of

$$\pi^i_k = \pi^i_{k-1}\prod_{j\in\mathcal{N}_i}\left(\frac{\pi^j_{k-1}}{\pi^i_{k-1}}\right)^\gamma \tag{8}$$

where $0 < \gamma < 1/\max(\{|\mathcal{N}_i| : i \in \mathcal{V}\})$.

If the supports of $\overline{\pi}^j_{k-1}$, $\forall j \in \mathcal{N}_i \cup \{i\}$ were totally overlapping (i.e. if these estimates were constructed from the same sample points) then one could simply compute $\overline{\pi}^i_k$ directly via (8). This is essentially the situation considered in Savic et al. (2012) and Lindberg et al. (2013) where the initial posteriors $\pi^i_{k=0}$ are sampled at the same point for each $i \in \mathcal{V}$. However, in the more likely case in which the supports of $\overline{\pi}^j_{k-1}$, $\forall j \in \mathcal{N}_i \cup \{i\}$ are totally disjoint (with probability 1) then an alternative method of computing an empirical estimate $\overline{\pi}^i_k$ of $\pi^i_k$ is needed.

Note that we obviously cannot sample directly from $\pi^i_k$ even if we know $\overline{\pi}^j_{k-1}$, $\forall j \in \mathcal{N}_i \cup \{i\}$. We can estimate $\pi^j_{k-1}$, $\forall j \in \mathcal{N}_i \cup \{i\}$ from $\overline{\pi}^j_{k-1}$, $\forall j \in \mathcal{N}_i \cup \{i\}$, using for example Kernel density estimation as in Silverman (1986), but we cannot then use this to compute an estimate of $\pi^i_k$ as no closed form solution to the update in (8) exists for any reasonable choice of the Kernel density estimate.

Instead we propose the following empirical approximation of $\pi^i_k$ based on importance sampling; see Doucet et al. (2001). Note

$$\begin{aligned}\pi^i_k &= \pi^i_{k-1}\prod_{j\in\mathcal{N}_i}\left(\frac{\pi^j_{k-1}}{\pi^i_{k-1}}\right)^\gamma \\ &= q^i_{k-1}\frac{\pi^i_{k-1}}{q^i_{k-1}}\prod_{j\in\mathcal{N}_i}\left(\frac{\pi^j_{k-1}}{\pi^i_{k-1}}\right)^\gamma\end{aligned} \tag{9}$$

or

$$\overline{\pi}^i_k(x) = \sum_{l=1}^{N_s}\frac{\pi^i_{k-1}(x^i_l)}{q^i_{k-1}(x^i_l)}\prod_{j\in\mathcal{N}_i}\left(\frac{\pi^j_{k-1}(x^i_l)}{\pi^i_{k-1}(x^i_l)}\right)^\gamma\delta(x - x_l) \tag{10}$$

where here $x_l \in \mathbb{R}^{m_x}$ are a set of $N_s$ independent and identically distributed samples of the so-called importance function $q^i_{k-1}$. Two matters remain. Firstly, one needs an importance function $q^i_{k-1}$ from which one can easily sample from. Secondly, we still don't know $\pi^j_{k-1}$, $\forall j \in \mathcal{N}_i \cup \{i\}$ and thus cannot compute $\pi^j_{k-1}(x^i_l)$, $\forall j \in \mathcal{N}_i \cup \{i\}$. To resolve the second matter we resort to Kernel density estimation and obtain

$$\overline{\pi}^i_k(x) = \sum_{l=1}^{N_s}\frac{\widetilde{\pi}^i_{k-1}(x^i_l)}{q^i_{k-1}(x^i_l)}\prod_{j\in\mathcal{N}_i}\left(\frac{\widetilde{\pi}^j_{k-1}(x^i_l)}{\widetilde{\pi}^i_{k-1}(x^i_l)}\right)^\gamma\delta(x - x^i_l) \tag{11}$$

where again $x_l \in \mathbb{R}^{m_x}$ are samples of $q^i_{k-1}$ and

$$\widetilde{\pi}^i_k(x) = \frac{1}{N_s}\sum_{\ell=1}^{N_s}\frac{1}{h}K\left(\frac{x - x^i_\ell}{h}\right) \tag{12}$$

where here $x_\ell \in \mathbb{R}^{m_x}$ are samples of $\pi^i_k$ or in other words correspond to the support of $\overline{\pi}^i_k(x)$ and thus are obviously known (i.e. by assumption at $k = 0$ we know $\overline{\pi}^i_k(x)$). The Kernel $K(\cdot)$ is chosen to be Gaussian in our simulations but other choices are possible; see Silverman (1986).

The importance function $q^i_{k-1}(x)$ must now be chosen and given the information available at the agents one obvious choice is

$$q^i_{k-1}(x) = \tilde{\pi}^i_{k-1}(x) \tag{13}$$

where the support of $\overline{\pi}^i_{k-1}(x)$ is distributed according to $q^i_{k-1}(x) = \tilde{\pi}^i_{k-1}(x)$ and so sampling from $q^i_{k-1}(x)$ is given. In this case

$$\widehat{\overline{\pi}}^i_k(x) = \sum_{\ell=1}^{N_s} w^i_{k,\ell}\delta(x - x^i_\ell) \tag{14}$$

where

$$w^i_{k,\ell} = \frac{\tilde{\pi}^i_{k-1}(x^i_\ell)}{\tilde{\pi}^i_{k-1}(x^i_\ell)}\prod_{j\in\mathcal{N}_i}\left(\frac{\tilde{\pi}^j_{k-1}(x^i_\ell)}{\tilde{\pi}^i_{k-1}(x^i_\ell)}\right)^\gamma \tag{15}$$

$$= \prod_{j\in\mathcal{N}_i}\left(\frac{\tilde{\pi}^j_{k-1}(x^i_\ell)}{\tilde{\pi}^i_{k-1}(x^i_\ell)}\right)^\gamma \tag{16}$$

with $w^i_{k,\ell} = w^i_{k,\ell}/\sum_{\ell=1}^{N_s} w^i_{k,\ell}$ and $x^i_\ell \in \mathbb{R}^{m_x}$ sampled from $q^i_{k-1}(x) = \tilde{\pi}^i_{k-1}(x)$ which in practice means $x^i_\ell$ are exactly the samples of $\overline{\pi}^i_{k-1}(x)$ at the previous time step. To combat possible degeneracy we then, given the $w^i_{k,\ell}$ and $x^i_\ell$ that define $\widehat{\overline{\pi}}^i_k(x)$, resample $N_s$ times from a multinomial distribution defined by $w^i_{k,\ell}$ and $x^i_\ell$ to obtain the final

$$\overline{\pi}^i_k(x) = \frac{1}{N_s}\sum_{\ell=1}^{N_s}\delta(x - x_\ell) \tag{17}$$

where now $x_\ell \in \mathbb{R}^{m_x}$ corresponds to the set of independent and identically distributed samples of the multinomial distribution just discussed.

*2.2 The Distributed Monte Carlo Fusion Algorithm*

From the perspective of sensor $i$ the Monte Carlo distributed fusion algorithm is:

**Step 0: Initialisation**
Pick $\gamma$ according to $0 < \gamma < 1/\max(\{|\mathcal{N}_i| : i \in V\})$, and

select a stopping time, $k_{\text{stop}}$. For $\ell = 1, \ldots, N_s$ sample $x^i_{k,\ell} \sim \pi^i_0$, where $\pi^i_0$ is a given initial probability density; e.g. $\pi^i_{k=0} \propto p(x)g(y_i|x)^n$ or $\pi^i_{k=0} \propto p(x)g(y_i|x)$.

**Step 1: Send, Receive** $(k > 0)$

For each $j \in \mathcal{N}_i$ send the unweighted samples set $\{x^i_{k,\ell}\}_{\ell=1}^{N_s}$ to agent $j$. For each $j \in \mathcal{N}_i$ receive $\{x^j_{k,\ell}\}_{\ell=1}^{N_s}$.

**Step 2: Calculate weights**

If $k < k_{\text{stop}}$ then for $\ell = 1 \ldots N_s$ calculate the fused particle weight $w^i_{k+1,\ell}$ via (15). If $k = k_{\text{stop}}$ then for $\ell = 1 \ldots N_s$ assign $w^i_{k+1,\ell} = \tilde{\pi}^i_k(x^i_{k,\ell})$.

**Step 3: Resample**

Resample with replacement $N_s$ points $x^i_{k+1,\ell}$ from the empirical density $\widehat{\pi}^i_{k+1}(x)$ in (14). This is equivalent to sampling from a multinomial distribution defined by $x^i_{k,\ell}$ and $w^i_{k+1,\ell}$. If $k = k_{\text{stop}}$ then the resampled set $\{x^i_{k+1,\ell}\}_{\ell=1}^{N_s}$ defines an unweighted empirical probability density of the form (17) which is the Monte Carlo approximation of either $Q \propto p(x)\prod_{i\in\mathcal{V}} g(y_i|x)^{1/n}$ or $Q' \propto p(x)\prod_{i\in\mathcal{V}} g(y_i|x)$ (depending on the definition of the initial $\pi^i_0$).

Note that we have restricted the number of iterations by the parameter $k_{\text{stop}}$. This is a practical approximation obviously and one could even use some defined threshold on the weights such that the weights in (15) approach one the algorithm halts. Again, depending on whether the initial likelihoods are conditionally independent or not, one may opt for convergence to $Q \propto p(x)\prod_{i\in\mathcal{V}} g(y_i|x)^{1/n}$ or $Q' \propto p(x)\prod_{i\in\mathcal{V}} g(y_i|x)$.

### *2.3 Illustrative Examples*

In this subsection we highlight the performance of the distributed Monte Carlo fusion algorithm initialised simply by $\pi^i_{k=0} \propto p(x)g(y_i|x)^n$ and where we seek convergence to the true optimal Bayesian fusion result $Q' \propto p(x)\prod_{i\in\mathcal{V}} g(y_i|x)$.

Firstly, we consider a random network with 5 agents shown in Figure 1. The true initial density $\pi^i_{k=0}$ at each agent was a randomly generated Gaussian mixture with 5 components. This initial density was then sampled at 1000 points to generate each agent's initial sampled density $\overline{\pi}^i_{k=0}(x)$ which all the agents would actually have access to in practice. The true initial densities at each agent and the corresponding random samples are shown in Figure 2.

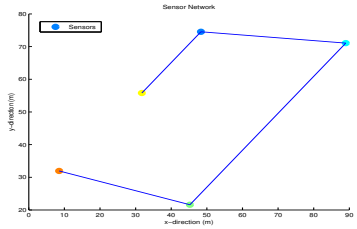


Fig. 1. The network topology with $n = 5$ agents.

The centralized Bayesian solution (which is computable from the true initial Gaussian mixtures) was computed for comparison and is shown in Figure 3. Note this cannot be computed in practice because the true initial (continuous) densities are unknown (only the samples are known) and

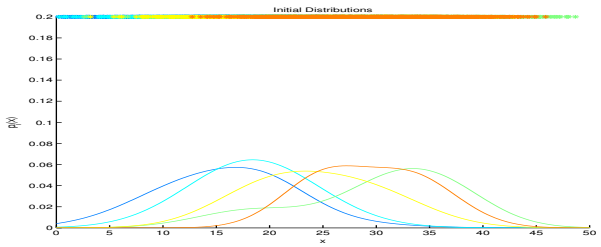


Fig. 2. The actual initial densities are Gaussian mixtures and the samples are shown above these for clarity.

of course we are interested in distributed computation in this work. However, this centralised solution is the ideal solution and represents the outcome we seek through iterative means and via Monte Carlo approximation.

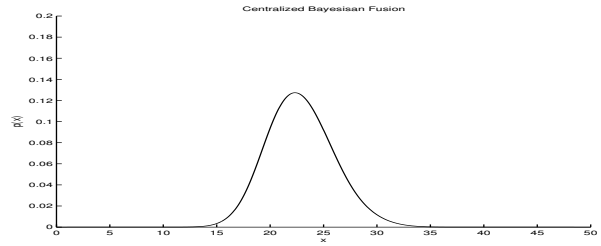


Fig. 3. The ideal centralised optimal Bayesian fusion result computed on the true underlying initial continuous densities. This is not computable in practice as we suppose only the empirical (sampled) density is initially known (and also the underlying true distributions are unlikely to Gaussian mixtures).

The centralised optimal Bayesian solution was compared to the distributed Monte Carlo algorithm for Bayesian fusion. The Monte Carlo solutions are shown (in colour) against the centralised solution (in black) in Figure 4.

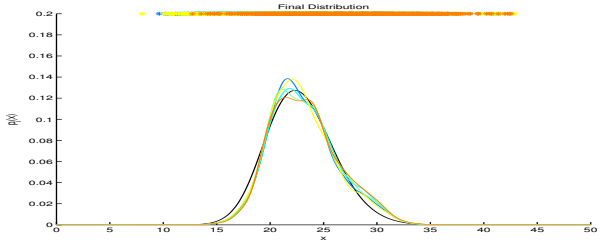


Fig. 4. The outcome from the distributed Monte Carlo fusion algorithm after 50 iterations. The empirical measure sample points are shown along with continuous Kernel density estimates generated from such (for visualisation only). The centralised optimal Bayesian solution shown in Figure 3 is also shown again here and we can see the Kernel estimates of the empirical (sampled) densities closely approximate the optimal centralised solution. They also have converged together and reached consensus.

From Figure 4 we clearly see that each agent has converged to a common value (i.e. they have reached consensus) and that this common value has converged to the desired (centralised Bayesian) solution.

We consider now a similar illustration with 10 agents and a network with cycles as shown in Figure 5. The true initial density $\pi^i_{k=0}$ at each agent was a randomly

generated Gaussian mixture with 5 components. This initial density was then sampled at 1000 points to generate each agent's initial sampled density $\overline{\pi}^i_{k=0}(x)$ which all the agent would actually have access to in practice. The true initial densities at each agent and the corresponding random samples are also shown in Figure 5.

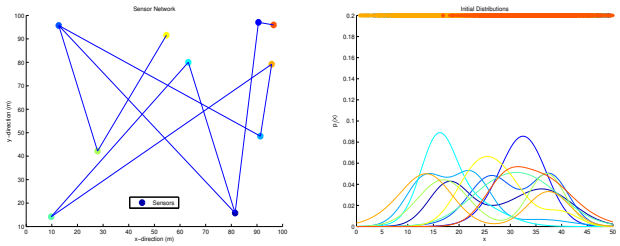


Fig. 5. The network topology with $n = 10$ agents and the actual initial densities are Gaussian mixtures and the samples are shown above these for clarity.

The centralized Bayesian solution (which is computable from the true initial Gaussian mixtures) was computed for comparison and is shown in Figure 6. Again this is just for comparison and is not realistically computable in practice.

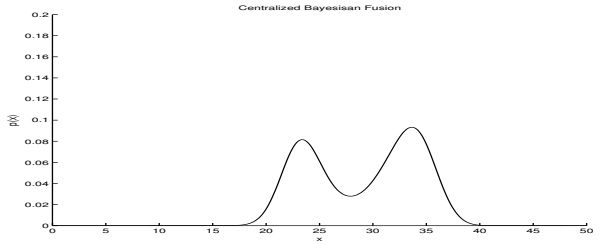


Fig. 6. The ideal centralised optimal Bayesian fusion result computed on the true underlying initial continuous densities. This is not typically computable in practice.

The centralised optimal Bayesian solution was compared to the distributed Monte Carlo algorithm for Bayesian fusion. The Monte Carlo solutions are shown (in colour) against the centralised solution (in black) in Figure 7.

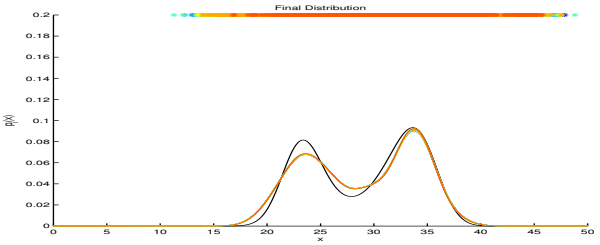


Fig. 7. The outcome from the distributed Monte Carlo fusion algorithm after 100 iterations. The empirical measure sample points are shown along with continuous Kernel density estimates generated from such (for visualisation only). The centralised optimal Bayesian solution shown in Figure 6 is also shown again here and we can see the Kernel estimates of the empirical (sampled) densities closely approximate the optimal centralised solution. They also have converged together and reached consensus.

From Figure 7 we clearly see that each agent has converged to a common value (i.e. they have reached consensus) and that this common value has converged closely to the desired (centralised Bayesian) solution. This multimodal fusion result shows the accuracy and potential of the distributed Monte Carlo fusion algorithm (as only the initial sample points in Fig 5 are used in initialising the algorithm and no knowledge about the underlying initial continuous densities is assumed).

## 3. A NOVEL METHOD FOR DISTRIBUTED PARTICLE FILTERING

In this section we apply the Monte Carlo data fusion algorithm to systems of networked particle filters. Particle filters are Bayesian filters that represent their posterior distributions by sets of unweighted samples.

The standard algorithm for particle filtering, also known as the bootstrap filter, works as follows:

### *3.1 The Bootstrap Filter Algorithm*

**Step 0: Initialisation** $(t = 0)$
For $\ell = 1 \dots N_s$ sample $x^i_{t,\ell} \sim p(x_0|x_0)$, where $p(x_0|x_0)$ is a known initial particle density.

**Step 1: Importance Sampling** $(t > 0)$
For $\ell = 1 \dots N_s$, propagate the samples forward in time through the update (system) equation, $x^i_{t,\ell} = f(x^i_{t-1,\ell}, u^i_{t,\ell})$ where $u^i_{t,\ell}$ is a sample from the distribution of process noise.

For $\ell = 1 \dots N_s$, evaluate the importance weights $w^i_{t,\ell} \propto g(y^i_t|x^i_t)$ and normalise, $\sum_{\ell=1}^{N_s} w^i_{t,\ell} = 1$.

**Step 2: Resample**
For $\ell = 1 \dots N_s$, sample $x^i_{t,\ell}$ from the multivariate distribution $\bar{p}(x^i_t|x^i_{t-1}, y^i_t)$ that is constructed from the samples $x^i_{t,\ell}$ and the weights $w^i_{t,\ell}$.

After resampling, the samples $\{x_{t,\ell}\}_{\ell=1}^{N_s}$ define the approximation of the posterior density of agent $i$. That is, we are approximating the true posterior given by

$$p(x_t|x_{t-1}, y^i_t) \propto p(x_t|x_{t-1}) g(y^i_t|x_t) \tag{18}$$

via a normalised empirical probability distribution defined by the samples $\{x_{t,\ell}\}_{\ell=1}^{N_s}$.

### *3.2 The Distributed Particle Filtering Algorithm*

Consider a group of agents (implementing individual particle filters) and an information sharing network defined by an undirected graph $\mathcal{G}$ as defined in Section 2.

The agents are tasked with estimating a target state. Each agent $i \in \mathcal{V}$, can make observations $y^i_t$ about the target at time $t$.

Our goal in distributed particle filtering is to exchange and combine posteriors such that each agent arrives at an approximation of the centralised posterior,

$$p(x_t|x_{t-1}, y^i_t) \propto p(x_t|x_{t-1}) \prod_{i \in V} g(y^i_t|x_t), \tag{19}$$

provided that the likelihood functions $g(y^i_t|x_t)$ for all $i \in \mathcal{V}$ are conditionally independent.

To achieve this goal we propose that the Monte Carlo fusion algorithm is run after Step 2 of the bootstrap algorithm.

From the perspective of sensor $i$:

**Step 0: Initialisation** $(t = 0)$
Pick $\gamma$ according to $0 < \gamma < 1/\max(\{|\mathcal{N}_j| : j \in V\})$. For $\ell = 1 \dots N_s$ sample $x^i_{t,\ell} \sim p(x^i_0|x^i_0)$, where $p(x^i_0|x^i_0)$ is a known initial particle density.

**Step 1: Importance Sampling** $(t > 0)$
For $\ell = 1 \dots N_s$, propagate the samples forward in time $x^i_{t,\ell} = f(x^i_{t-1,\ell}, u^i_{t,\ell})$ where $u^i_{t,\ell}$ is a sample from the distribution of process noise.

For $\ell = 1 \dots N_s$, evaluate the importance weights $w^i_{t,\ell} \propto g(y^i_t|x^i_t)^{|V|}$ and normalise.

**Step 2: Resample**
For $\ell = 1 \dots N_s$, sample $x^i_{t,\ell}$ from the multinomial distribution $\bar{p}$ constructed from the samples $\{x^i_{t,\ell}\}_{\ell=1}^{N_s}$ and the weights $\{w^i_{t,\ell}\}_{\ell=1}^{N_s}$.

**Step 3: Distributed Fusion**
Run fusion algorithm of Section 2.2 for $k = 0 \dots k_{stop}$. For $\ell = 1 \dots N_s$ assign $x^i_{t,\ell} = x^i_{k_{stop},\ell}$ and $w^i_{t,\ell} = w^i_{k_{stop},\ell}$. Note that the weights should be equal to $1/N_s$ as the samples $x^i_{k_{stop},\ell}$ should be unweighted.

### *3.3 Illustrative Examples*

In this subsection we demonstrate the distributed particle filtering algorithm on the benchmark nonlinear dynamical system of Gordon et al. (1993). The process equation of this system is

$$x_t = 0.5x_{t-1} + \frac{25x_{t-1}}{1 + {x_{t-1}}^2} + 8\cos(1.2(t-1)) + u_t, \quad (20)$$

where $u_t$ is Gaussian white noise with variance, $\sigma^2 = 10$; and the sensor equation is

$$y^i_t = \frac{{x_t}^2}{20} + v^i_t, \quad (21)$$

where $v^i_t$ is Gaussian white noise with a random variance. We initialised the filters with the state $x^i_0 = 0.1$ for all $i \in V$. This represents a system with a known initial state.

The first simulation was conducted over a network of 10 agents, see Figure 8, with $N_s = 1500$ samples. The true state of the system is also shown in Figure 8. We use the version of the fusion algorithm that converges to $Q'$ (see Corollary 2) as the likelihood functions are known to be conditionally independent. Therefore we anticipate that the fusion algorithm will converge to the centralised posterior density.

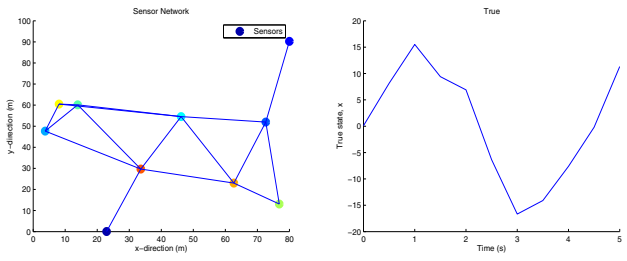


Fig. 8. The network topology of the second distributed particle filter simulation with $n = 10$ agents and the true state of the nonlinear dynamical system which the agents are trying to estimate.

Between measurements each agent runs the distributed fusion algorithm with $k_{stop} = 15$. In Figure 9 we compare the resulting estimates (which we call the fusion estimates) to the desired state and the centralised estimate (computed at a hypothetical agent with access to each agent's likelihood functions). The fusion estimates are shown by the coloured dotted lines while the centralised estimate is shown in solid blue and the true state in solid black. It is clear that after fusion the agents arrive at estimates that are close to the centralised solution. In Figure 10 we show the estimates that would be obtained if each node ran the bootstrap filter algorithm in isolation (i.e. using only their individual local likelihood functions). We refer to these as the isolated estimates as they do not involve any communication (exchange of information). By comparison to the fusion estimates, the isolated estimates are scattered around the centralised solution as expected.

In Figure 11 we show the estimates generated by a local particle filter running at each agent that makes use of the just the independent likelihood functions from each agent's neighbours. We refer to these as the local estimates and this method. The local estimate of $i$ can be thought of as centralised estimate in the neighbourhood of agent $i$. In a complete network the local estimates are equivalent to the centralised estimates.

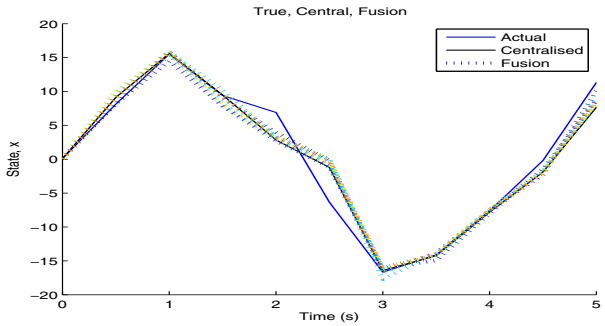


Fig. 9. The estimates made by the distributed particle filtering algorithm (coloured dotted lines). These are shown with the centralised estimates (solid blue) and the true state (solid black). The figure shows that the agents are approximately reaching a consensus about the fused posterior near the centralised solution as desired.

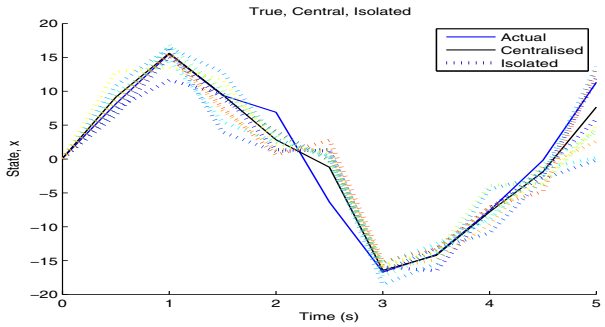


Fig. 10. The isolated estimates (coloured dotted lines) are shown along with the centralised estimate and the true state. As expected the centralised estimate falls between the isolated estimates which are computed by the standard bootstrap filter algorithm.

In the second example we randomly generate another network of 10 agents and a new underlying system trajectory;

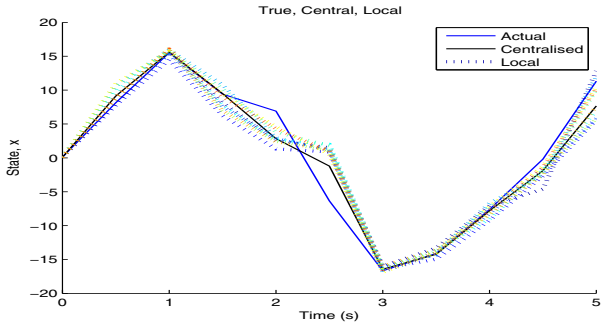


Fig. 11. The local estimates (coloured dotted lines) are shown along with the centralised estimate and the true state. The local estimates are made by exchanging likelihoods between neighbours only. This is important as a benchmark for our proposed algorithm as this is the simplest algorithm that may be implemented in a network of particle filters.

see Figure 12. Again the likelihood functions are conditionally independent, however we unnecessarily use the conservative fusion method (for demonstration purposes). As the observations are independent this method leads to sub-optimal estimates.

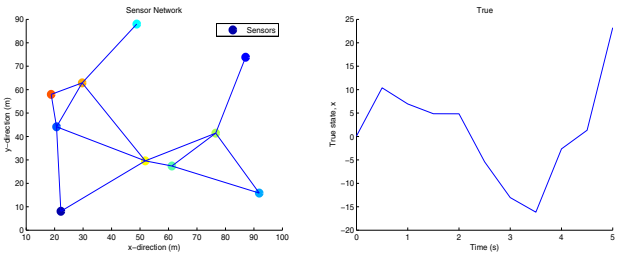


Fig. 12. The network topology of the second distributed particle filter simulation with $n = 10$ agents and the true state of the nonlinear dynamical system which the agents are trying to estimate.

The agents run the conservative distributed fusion algorithm with $k_{stop} = 20$ which converges towards the consensus value $Q \propto p(x) \prod_{j \in V} g(y_j|x)^{1/n}$. This is the conservative posterior. The fusion estimates were compared to the true state and the centralised estimate in Figure 13. We can see that the agents are approaching a common value (i.e. they are close to consensus).

In Figures 14 and 15 we show the isolated and local estimates respectively (see the coloured dotted lines). It can be seen that although the local estimates are an improvement on the isolated estimates, the conservative fusion estimates are better again. This simulation shows that it is possible to make useful state estimates even when the dependence between the agents is unknown [Bailey et al. (2012)].

## 4. CONCLUSION

We presented a practical method for distributed data fusion and particle filtering. The algorithm is consensus-based and involves the exchange of posteriors between neighbours. The proposed solution is robust to time-varying network topologies including those with cycles and is guaranteed consistent.

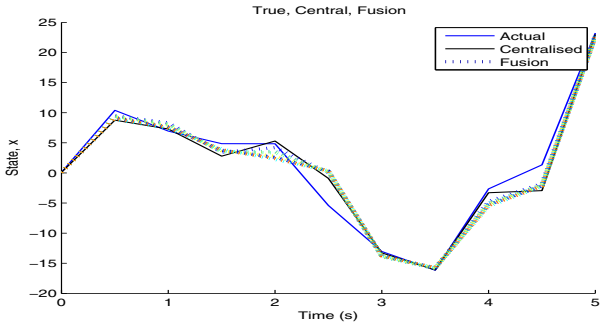


Fig. 13. The estimates made by the distributed particle filtering algorithm (coloured dotted lines). These are shown with the centralised estimates (solid blue) and the true state (solid black). The figure shows that the agents are approximately reaching a consensus about the fused posterior near the centralised solution as desired.

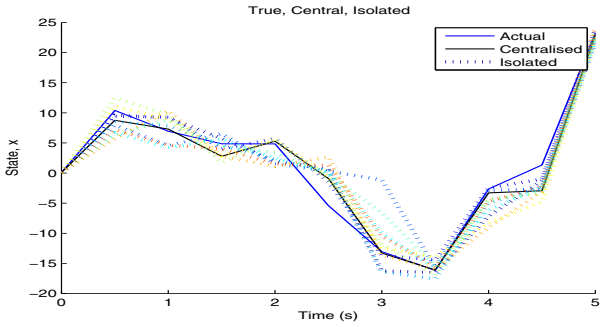


Fig. 14. The isolated estimates (coloured dotted lines) are shown along with the centralised estimate and the true state. As expected the centralised estimate falls between the isolated estimates which are computed by the standard bootstrap filter algorithm.

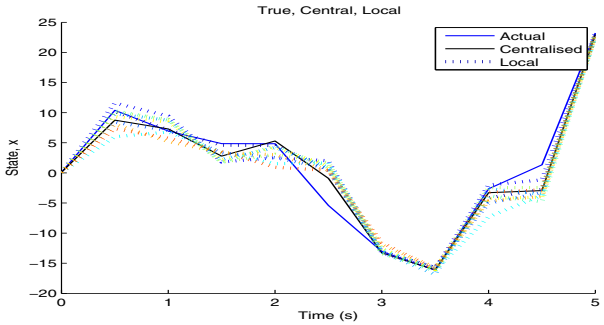


Fig. 15. The local estimates (coloured dotted lines) are shown along with the centralised estimate and the true state. The local estimates are made by exchanging likelihoods between neighbours only.

In the future we would like to experiment with other consensus algorithms such as the dynamic consensus algorithm of Zhu and Martínez (2010) to reduce the communication overhead.

## REFERENCES

Abbas, A. (2009). A Kullback-Leibler view of linear and log-linear pools. *Decision Analysis*, 6(1), 25–37.

Bailey, T., Julier, S., and Agamennoni, G. (2012). On conservative fusion of information with unknown non-gaussian dependence. In *15th International Conference*

*on Information Fusion (FUSION)*, 1876–1883. Singapore.

Bashi, A.S., Jilkov, V.P., Li, X.R., and Chen, H. (2003). Distributed implementations of particle filters. In *Proceedings of the Sixth International Conference of Information Fusion*, volume 2, 1164–1171. Cairns, Australia.

Bolić, M., Djurić, P.M., and Hong, S. (2005). Resampling algorithms and architectures for distributed particle filters. *IEEE Transactions on Signal Processing*, 53(7), 2442–2450.

Doucet, A., de Freitas, N., and Gordon, N. (eds.) (2001). *Sequential Monte Carlo Methods in Practice*. Springer.

Farahmand, S., Roumeliotis, S.I., and Giannakis, G.B. (2011). Set-membership constrained particle filter: Distributed adaptation for sensor networks. *IEEE Transactions on Signal Processing*, 59(9), 4122–4138.

Gordon, N., Salmond, D., and Smith, A.F.M. (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2), 107–113.

Gu, D. (2007). Distributed particle filter for target tracking. In *IEEE International Conference on Robotics and Automation*, 3856–3861. Rome.

Gu, D., Sun, J., Hu, Z., and Li, H. (2008). Consensus based distributed particle filter in sensor networks. In *International Conference on Information and Automation (ICIA)*, 302–307. Changsha.

Hlinka, O., Djurić, P.M., and Hlawatsch, F. (2009). Time-space-sequential distributed particle filtering with low-rate communications. In *Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, 196–200. Pacific Grove, CA.

Hlinka, O., Hlawatsch, F., and Djurić, P.M. (2013). Distributed particle filtering in agent networks: A survey, classification, and comparison. *IEEE Signal Processing Magazine*, 30(1), 61–68.

Hlinka, O., Slučiak, O., Hlawatsch, F., Djurić, P.M., and Rupp, M. (2010). Likelihood consensus: Principles and application to distributed particle filtering. In *Conference Record of the Forty Fourth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, 349–353. Pacific Grove, CA.

Hlinka, O., Slučiak, O., Hlawatsch, F., Djurić, P.M., and Rupp, M. (2011). Distributed gaussian particle filtering using likelihood consensus. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3756–3759. Prague.

Hlinka, O., Slučiak, O., Hlawatsch, F., Djurić, P.M., and Rupp, M. (2012). Likelihood consensus and its application to distributed particle filtering. *IEEE Transactions on Signal Processing*, 60(8), 4334–4349.

Lee, S.H. and West, M. (2009). Markov chain distributed particle filters (mcdpf). In *Proceedings of the 48th IEEE Conference on Decision and Control (CDC) and 28th Chinese Control Conference*, 5496–5501. Shanghai.

Lee, S.H. and West, M. (2010). Performance comparison of the distributed extended kalman filter and markov chain distributed particle filter (mcdpf). In *Proceedings of the 2nd IFAC Workshop on Distributed Estimation and Control in Networked Systems (NecSys'10)*, 151–156. Annecy, France.

Lee, S.H. and West, M. (2013). Convergence of the markov chain distributed particle filter (mcdpf). *IEEE Transactions on Signal Processing*, 61(4), 801–812.

Lindberg, C., Muppirisetty, L., Dahlén, K.M., Savic, V., and Wymeersch, H. (2013). Mac delay in belief consensus for distributed tracking. In *Proc. of 10th IEEE Workshop on Positioning, Navigation and Communication*. Dresden, Germany.

Mohammadi, A. and Asif, A. (2009). Distributed particle filtering for large scale dynamical systems. In *IEEE 13th International Multitopic Conference (INMIC)*, 1–5. Islamabad.

Mohammadi, A. and Asif, A. (2011a). Consensus-based distributed unscented particle filter. In *IEEE Statistical Signal Processing Workshop (SSP)*, 237–240. Nice, France.

Mohammadi, A. and Asif, A. (2011b). A consensus/fusion based distributed implementation of the particle filter. In *4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 285–288. San Juan.

Olfati-Saber, R., Franco, E., Frazzoli, E., and Shamma, J.S. (2006). Belief consensus and distributed hypothesis testing in sensor networks. In *Network Embedded Sensing and Control. (Proceedings of NESC05 Workshop), volume 331 of Lecture Notes in Control and Information Sciences*, 169–182. Springer Verlag.

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

Oreshkin, B.N. and Coates, M.J. (2010). Asynchronous distributed particle filter via decentralized evaluation of gaussian products. In *13th Conference on Information Fusion (FUSION)*, 1–8. Edinburgh.

Savic, V., Wymeersch, H., and Zazo, S. (2012). Distributed target tracking based on belief propagation consensus. In *Proceedings of the 20th European Signal Processing Conference*, 544–548. Bucharest, Romania.

Savic, V., Wymeersch, H., and Zazo, S. (2012). Distributed target tracking based on belief propagation consensus. In *Proceedings of the 20th European Signal Processing Conference (EUSIPCO)*, 544–548. Bucharest.

Sheng, X. and Hu, Y.H. (2005). Distributed particle filters for wireless sensor network target tracking. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 4, 845–848.

Sheng, X., Hu, Y.H., and Ramanathan, P. (2005). Distributed particle filter with gmm approximation for multiple targets localization and tracking in wireless sensor network. In *Fourth International Symposium on Information Processing in Sensor Networks (IPSN)*, 181–188.

Silverman, B.W. (1986). *Density Estimation for Statistics and Data Analysis*. Monographs on Statistics and Applied Probability. Chapman and Hall.

Üstebay, D., Coates, M., and Rabbat, M. (2011). Distributed auxiliary particle filters using selective gossip. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3296–3299. Prague.

Zhu, M. and Martínez, S. (2010). Discrete-time dynamic average consensus. *Automatica*, 46(2), 322–329.