Resampling and Network Theory

Praveen B. Choppala, *Member, IEEE*, Marcus R. Frean, and Paul D. Teal, *Senior Member, IEEE*

***Abstract*—Particle filtering provides an approximate representation of a tracked posterior density which converges asymptotically to the true posterior as the number of particles used increases. The greater the number of particles, the higher the computational complexity. This complexity can be implemented by operating the particle filter in parallel architectures. However, the resampling step in the particle filter requires a high level of synchronization and extensive information interchange between the particles, which impedes the use of parallel hardware systems. This paper establishes a new perspective for understanding particle filtering — that particle filtering can be achieved by adopting the principles of information exchange within a network, the nodes of which are now the particles in the particle filter. We propose to connect particles via a minimally connected network and resample each locally. This strategy facilitates full information exchange among the particles, but with each particle communicating with only a small fixed set of other particles, thus leading to minimal communication overhead. The key benefit is that this approach facilitates the use of many particles for accurate posterior approximation and tracking accuracy.**

***Index Terms*—Particle filter, resampling, networks, greedy, stochastic, Kolmogorov-Smirnov statistic.**

I. Introduction

SEQUENTIAL Bayesian filtering is a powerful tool for online estimation of the states of systems and involves the recursive computation of the posterior probability density function (PDF) of the state using the previous information and the sensor evidence. Bayesian filtering for nonlinear non-Gaussian applications involves high dimensional integration that renders the filtering process intractable. This intractability is overcome using approximate solutions, and the approximation that is of key interest to this paper is the particle filter (PF) [1]–[3]. The PF approximates the posterior PDF of the target state by a set of weighted particles. The first stage in the PF is sequential importance sampling (SIS) that describes the process of drawing new particles at each time step and updating a corresponding weight for each. By itself, SIS encounters the degeneracy problem in which, after a few iterations, the weight is distributed over very few particles, causing the filter to devote large computational effort to update particles that do not contribute to the posterior. This problem is overcome in the second stage, *resampling* [2], [4], [5]. The focus of this paper is on resampling, the purpose of which is to eliminate those particles that have low weights and replace them by copies of other particles that have large weights, while ensuring that the resampled particle set accurately approximates the posterior.

The proposed resampling method is considered here in the context of particle filter tracking, it can be applied to a wide class of applications including biology [6], neural networks [7] and database management [8].

The PF parallelization problem: A key feature of the PF is that the weighted particle approximation will approach the true posterior if a large number of particles are used. However the use of more particles for an accurate result is constrained by high computational complexity [9] and hence PFs are computationally demanding for real time applications. This complexity can be conveniently overcome by the use of graphical processing units (GPUs) and field programmable gate arrays (FPGAs) that aid in massive data parallelism and pipelining of operations to maximize the throughput [10]. While the SIS stage of the PF can be readily parallelized because particle prediction and update do not require particle concurrency, the major bottleneck to the parallelization of the PF is the resampling step. Resampling imposes the following constraints; a) synchronization among all the particles leading to particle inter-dependence, b) extensive data exchange between the particles, and c) continuous re-routing of particles, which we term volatility, i.e., if each particle is considered to be processed by a single processing unit, then the "wiring diagram" between the processing units changes at every time step. These constraints prohibit the use of parallel hardware systems for the PF. Hence to date, there has been very limited implementation of the PF in hardware for real-time industrial use.

A. PF Parallelization: Related Work and Challenges

The work on particle filter parallelization, that started in the early 2 000s [11], branched into two lines of thought: a) architectural development, and b) algorithmic development. The main idea in architectural development is to either centrally (e.g., [11], [12]) or locally (e.g., [9], [13]–[15]) resample particles that are distributed across several processing elements (PEs) operating in parallel. In the centralized architectures, each PE is assigned a fixed number of particles and it completes all sequential operations on its particles. A central unit then receives the particles from all the PEs and performs resampling. The speed improvement in these methods is minimal because only the sequential operations of the particle filter are parallelized. The exchange of weights and indices between the central unit and the PEs is deterministic, but the particle routing is still volatile because

Manuscript received March 28, 2020; revised August 9, 2021, November 18, 2021, and January 14, 2022; accepted January 18, 2022. Date of publication January 25, 2022; date of current version February 22, 2022. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Tirza Routtenberg. *(Corresponding author: Praveen B. Choppala.)*

Praveen B. Choppala is with the Department of E.C.E., WISTM, Andhra University, Visakhapatnam, Andhra Pradesh 530003, India (e-mail: praveenbchoppala@gmail.com).

Marcus R. Frean and Paul D. Teal are with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6012, New Zealand (e-mail: marcus.frean@ecs.vuw.ac.nz; paul.teal@vuw.ac.nz).

Digital Object Identifier 10.1109/TSIPN.2022.3146051

there is no prior knowledge about which particles will be shared among the PEs. In the local (or distributed) architectures, each PE performs both sequential operations and local resampling on each of its particles and then the residuals (deficient or additional particles), if any, are exchanged between the other processing elements via resampling at a central processing unit. This reduces data exchange between the processing elements and the number of particles processed by the central processing unit. A further reduction in the particle interaction is achieved in the recently proposed distributed computing PF [16], in which each PE shares a few of its higher weight particles with its neighboring PE via a ring topology. Methods that accelerate the particle filter operation within the processing elements by exploiting some features inherent to graphical processing units have also been proposed, for example, see [17]. On the whole, the emphasis of these methods is on distributed parallelism. A key limitation in these methods is that the particles to be shared among the processing elements are not known in advance, therefore the connectivity between particles varies at each time step. Therefore these methods are not fully parallel-friendly.

Regarding the algorithmic development approach: the most conventionally used stochastic resamplers are the multinomial [18], the stratified [19], the residual [20], the systematic [2], [21] the residual systematic [9] and the soft systematic [22] resampling. These stochastic resamplers demand extensive and ever-changing inter-particle interaction during; a) normalization, b) evaluation of the cumulative sum of the weights, and c) searching through all the particles to find a particle, the cumulative weight of which is greater than a sample drawn from the uniform density $\mathcal{U}(0,1]$. The properties of resampling and comparison are presented in detail in [4], [5]. The methodology of these resampling schemes involves intensive communication within the particles, rendering them unsuitable for parallel processing systems. The deterministic resamplers (namely proportional redistribution [12] and partial deterministic [23] resampling) alleviate this problem by resampling those particles whose weights are larger or smaller than suitably chosen thresholds. The performance of this approach is sensitive to the choice of these thresholds and leads to some loss of information contained in discarded particles.

The recently proposed Metropolis [24], [25] and the coalesced Metropolis [26] resamplers propose that each particle obtains information from a few other particles, contrary to the *all* particles required in the traditional case, and a selection process is applied thereafter to sample one particle to replace the current particle. Although this reduces particle synchronization, particle interchange is still volatile because there is no prior knowledge of the set of particles with which each particle must be compared. Other proposals involving Metropolis Hastings sampling schemes have shown only limited improvement [10]. A hardware implementation scheme for the popularly used systematic resampler was proposed in the very recent parallel systematic/stratified resampler [27], but the method still suffers from ever-changing inter-particle connectivity throughout the filtering process. Other recent developments in algorithm development in conjunction with parallel architectures include [28]–[30].

B. Contribution of This Paper

This paper presents a resampling technique which does not require the substantial communication overhead suffered by the conventional resamplers. The key idea here is to adopt the principles of information exchange within nodes in a network to information exchange among particles in the particle filter. Our motivating rationale is grounded on sound principles of network theory: treating particles in a PF as nodes in a network provides us with a mathematical framework to minimize communication and maximize information within the particles, analogous to communication in small-world networks [31]. This paper proposes that by bringing the principles of network theory into the PF context, the particles can communicate with their neighbors as do the nodes in a network, and accomplish resampling with minimal inter-particle communication and yet complete information transfer. Consequently the filter may operate on millions of particles but with minimal communication within them. The contributions of this paper are two-fold, (a) the promulgation of the use of principles of network theory in particle filtering, and (b) the proposal of greedy and stochastic variants of network based resampling. This work will lead to a phenomenal speed-up of particle filtering by virtue of substantially reducing the communication overhead within the particles and provide a step ahead in the community's pursuit to develop efficient, accurate and flexible tracking methods.

The rest of the paper is organized as follows: Section II sets the notation and outlines sequential Bayesian estimation. After introducing a few networks in section III, the applicability of network theory to resampling is proposed in section IV. This is followed by evaluation results in section V and concluding remarks in section VI.

II. Sequential Bayesian Estimation

In this section, sequential Bayesian estimation for target tracking and its approximation using a PF [1] is presented. The state of each target $\mathbf{x}_t$ at time t may be characterized by the parameters that define it; for example, its position, velocity, acceleration etc. The target maneuvers are modelled as Markovian motion and the aim is to recursively estimate $\mathbf{x}_t$ using the noisy observations $\mathbf{y}_{1:t} = \{\mathbf{y}\}_{i=1}^{t}$ received from the sensors until the tth time step. Bayesian filtering for this estimation is a two step procedure. If the distribution pdf at time $t-1$ is available and is given by $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$, then the posterior pdf at time t, expressed as $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ can be obtained by prediction and update as

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})d\mathbf{x}_{t-1} \quad (1)$$

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})} \quad (2)$$

The integrals in (1) and implicit in the denominator of (2) are high dimensional entities and hence intractable. This intractability can be overcome using an approximation technique — the PF, in which the posterior $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ is represented by a set of particles and their associated weights as $\{\mathbf{x}_{t-1}^i, w_{t-1}^i\}_{i=1}^N$, where i is the particle index and N is the total number of particles. In

the prediction step, a new set of particles are sampled from an importance distribution as

$$\bar{\mathbf{x}}_t^i \sim q(\mathbf{x}_t^i|\mathbf{x}_{t-1}^i, \mathbf{y}_{1:t}) \tag{3}$$

and their weights are updated as

$$\bar{w}_t^i = w_{t-1}^i \frac{p(\mathbf{y}_t|\bar{\mathbf{x}}_t^i)\, p(\bar{\mathbf{x}}_t^i|\mathbf{x}_{t-1}^i)}{q(\bar{\mathbf{x}}_t^i|\mathbf{x}_{t-1}^i, \mathbf{y}_{1:t})} \tag{4}$$

and then normalised. The time recursion of the prediction and the update steps forms the basis for the PF. However, after a few iterations, the discrepancy between the weights increases, leading to degeneracy. A solution to this is resampling, that replicates particles based on their weights so that the particles having negligible weights are eliminated, and replaced by those with larger weights [32]. The representation of the resampled posterior is

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \sum_{i=1}^{N} \bar{w}_t^i \delta(\mathbf{x}_t - \bar{\mathbf{x}}_t^i) = \sum_{i=1}^{N} \frac{n_t^i}{N} \delta(\mathbf{x}_t - \bar{\mathbf{x}}_t^i) \tag{5}$$

where n_t^i denotes the number of replications of the ith particle determined in accordance to its weight. Conventional resampling methods that give an unbiased representation of the posterior [4], [5] obtain a new weighted set of particles $\{\bar{\mathbf{x}}_t^i, \bar{w}_t^i\}_{i=1}^N \longrightarrow \{\mathbf{x}_t^i, w_t^i\}_{i=1}^N$ as follows: for $i = 1, \ldots, N$, we sample an index $j(i)$ distributed according to the probability $\mathrm{P}(j(i) = m) = \bar{w}_t^m, m = 1, \ldots, N$ and assign $\mathbf{x}_t^i = \bar{\mathbf{x}}_t^{j(i)}$ and set $w_t^i = 1/N$. Then (5) changes to

$$\hat{p}(\mathbf{x}_t|\mathbf{y}_{1:t}) \approx \sum_{i=1}^{N} w_t^i \delta(\mathbf{x}_t - \mathbf{x}_t^i) = \sum_{i=1}^{N} \frac{1}{N} \delta(\mathbf{x}_t - \mathbf{x}_t^i) \tag{6}$$

This resampling operation, being sequential and also exhaustive in nature (requiring information on all the particles) renders the PF computationally expensive. In the sequel we propose a resampling strategy using graph structures that will substantially speed up the PF resampling process.

III. Network Theory

This section briefly introduces three common network structures and presents the procedure to construct them. The use of these networks in the context of PF resampling is an original contribution made by this paper. Mathematically, a graph is an ordered pair $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ is the set of edges. If there is a direct connection between two nodes, then the nodes are termed neighbours. The connectivity that a network offers with only a few neighbours is the property that defines the sparsity of the graph. The adjacency matrix $\mathbf{G}$ for a directed but unweighted graph is defined by

$$\mathbf{G}_{i,j} = \begin{cases} 1 & \text{if } \mathcal{V}_i \to \mathcal{V}_j \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where $\mathcal{V}_i$ is the ith node, $\mathcal{V}_j$ is the jth node and $\to$ symbolizes that a directed edge exists from $\mathcal{V}_i$ to $\mathcal{V}_j$.

The *degree* of a node is the number J of its neighbours. In general the degree will differ for each node of a network. However, in this paper, nodes represent processing elements, and for implementation reasons it is desirable to keep these as similar as possible. Hence we only consider networks that have fixed degree.

For a sparse network most of the entries of $\mathbf{G}$ are zero. Consider a network containing N nodes. The goal is to find J neighbours for each node such that the information on any node can be easily transmitted across all the nodes in the minimum possible number of hops. Some well known networks used in this context are now presented. (Some other network structures [33] including those with bus, the square, the ring or the star topologies, and the scale free networks [34] in general do not offer high connectivity, and a circuitous route is required for the information in a node to reach every other node.)

A. Recursive Binary Partitioning Network

The recursive binary partitioning (BP) network [35] aims to deterministically transmit the information contained in any node to any other node in logarithmic time, i.e., for N nodes, it requires only $\log_2 N$ time steps and creates only $J = \log_2 N$ edges to each node. This network is formed by recursively partitioning the N nodes into subsets containing $2\mathbf{n}_p$ nodes each while every node connects to the $\mathbf{n}_p$-th node in that subset. It is evident that the number of nodes N in this scheme can only be powers of 2. An example adjacency matrix $\mathbf{G}$ for the recursive binary partitioning network is shown in Fig. 1(a).

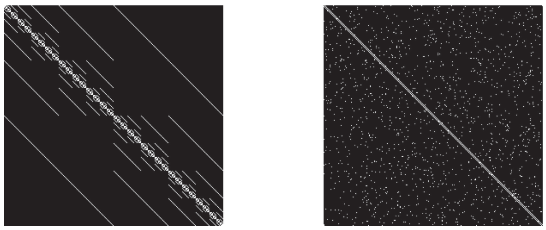
B. Small-World Network

A small-world (SW) network is a network in which no node has a large number of links yet the information in any node can reach any other node by hopping through only a few intermediary nodes. One class of random network that exhibits small-world network properties is the *Watts-Strogatz model* [36]. For a fixed-degree SW network, the J neighbours to each node are formed as follows: firstly form a ring topology among the N nodes such that every node has 2 neighbours. Then for each node, create $J - 2$ edges to other nodes that are not already connected to the node under examination, which are chosen at random with uniform probability. An example of $\mathbf{G}$ for a Watts-Strogatz network is shown in Fig. 1(b).

C. Fixed-Degree Random Networks

In a Erdős-Rényi network [37] each node is connected to any other with equal probability p. As for the previous two networks, we again restrict each node to have degree J, which is a special case of what are known as random graphs with fixed degree sequence. The larger the value of J for this fixed-degree random (FDR) network, the faster the transmission of information among nodes is likely to be. Each node is considered to be connected to itself. Small values of J (e.g., $J < 2$) could result in islands of nodes such that the information contained in a node cannot reach every node, although it has been shown that the probability of an island is very low for $J \geq \log N$ [37]–[39].

The property of these networks to achieve complete information transfer within nodes with minimal connectivity and communication overhead is our motivation for their use in the PF



(a) Recursive binary partitioning. (b) Small-world (Watts-Strogatz).

Fig. 1. The adjacency matrix $\mathbf{G}$ constructed from (7). The number of nodes is $N = 256$ and the network connectivity is $J = \log_2(N) = 8$. The FDR network is visually similar to (b). The black portions of the figures represent zeros and the white portions represent ones. When J is small the density of ones (white portion) is minimal.

TABLE I
THE GRAPH RADIUS FOR THE FIXED DEGREE RANDOM (FDR), SMALL-WORLD (SW) AND RECURSIVE BINARY PARTITIONING (RB) NETWORKS. SINCE THE FDR AND SW NETWORKS ARE CONSTRUCTED STOCHASTICALLY, THEIR RESULTS ARE AVERAGED OVER 100 NETWORK REALISATIONS

N	64			1024			4096		
J	FDR	SW	RB	FDR	SW	RB	FDR	SW	RB
3	5.9	32	-	9	512	-	10.2	2048	-
5	3.9	4.0	-	5.4	6.05	-	6	7	-
6	3.0	3.8	6	5	5.1	-	6	6	-
10	-	-	-	4	4	10	4.5	5	-
12	-	-	-	-	-	-	4	4	12

context. The length of the shortest path from any node to every other node (the graph radius) is shown in Table I for different N and J. It can be observed that at low connectivity $J = 3$, the FDR network has the shortest path amongst the three networks. At $J = \log_2(N)$ the FDR and SW networks have graph radius of four hops whereas the graph radius for the recursive binary partitioning network is $\log_2(N)$.

IV. RESAMPLING AND NETWORK THEORY

In this section, the proposal to resample particles using the above mentioned networks is presented and its merits are then discussed.

A. The Methodology

Let $\{\mathbf{x}_{t-1}^i, w_{t-1}^i\}_{i=1}^N$ be the set of support points, or weighted particles, at time $t-1$, that represent the posterior pdf $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$. A single time-step recursion from $t-1 \longrightarrow t$ of the PF generates a new set of weighted particles in two steps,

$$\{\mathbf{x}_{t-1}^i, w_{t-1}^i\}_{i=1}^N \longrightarrow \underbrace{\{\bar{\mathbf{x}}_t^i, \bar{\mathbf{w}}_t^i\}_{i=1}^N}_{\text{Imp. Sampling}} \longrightarrow \underbrace{\{\mathbf{x}_t^i, w_t^i\}_{i=1}^N}_{\text{Resampling}}$$

such that support set $\{\bar{\mathbf{x}}_t^i, \bar{\mathbf{w}}_t^i\}_{i=1}^N$ is obtained from importance sampling according to (3) and (4) and the posterior pdf at time t is represented by (5) after normalising the weights. The implicit degeneracy problem in this set is overcome by resampling a new support set $\{\mathbf{x}_t^i, w_t^i\}_{i=1}^N$ such that

$$\mathrm{P}(\mathbf{x}_t^i \longleftarrow \bar{\mathbf{x}}_t^j) = \bar{w}_t^j, \qquad i = 1, \ldots, N \tag{8}$$

The conventional approach to achieve this is via multinomial selection. Here, we first generate independent and identically distributed (i.i.d.) random samples $u^i \sim \mathcal{U}(0,1), i = 1, \ldots, N$ and then set the index $I(i)$ to the index of the j-th value of the cumulative sum, i.e.,

$$I(i) = j \ : \ u_i \in \left(\sum_{s=1}^{j-1} \bar{w}_t^s, \sum_{s=1}^{j} \bar{w}_t^s \right), \quad i = 1, \ldots, N \tag{9}$$

and then assign

$$\mathbf{x}_t^i = \bar{\mathbf{x}}_t^{I(i)}, \quad w_t^i = 1/N, \quad i = 1, \ldots, N \tag{10}$$

This procedure is equivalent to sequentially climbing the cumulative sum "ladder" $\sum_{s=1}^{j} \bar{w}_t^s$ up to the jth sample that satisfies the above condition. This search is performed for all the $i \in \{1, \ldots, N\}$ particles sequentially. The key disadvantages in this approach is that the length of the climb for each particle is unknown and can possibly reach N steps. The stratified and systematic resamplers reduce the length of the climb by sampling u^i from the equally spaced strata in (0,1] space. Nevertheless this climb is sequential and the entire operation over $i \in \{1, \ldots, N\}$ means that the maximum order of complexity for one time step for this resampling will be $\mathcal{O}(N^2)$.

This paper proposes to treat a node as a particle and an edge between two nodes implies that electronic wiring connects the processing hardware of two particles. For N particles (or nodes), one can form a network using any of the methods described in Section III such that $\mathcal{V}_i \rightarrow \mathcal{V}_j \Rightarrow \bar{\mathbf{x}}_t^i \rightarrow \bar{\mathbf{x}}_t^j$. It is clear that for low values of connectivity J, the adjacency matrix $\mathbf{G}$ is very sparse and therefore the connections between the particles are minimal. In our proposed implementation $\mathbf{G}$ is computed only once and remains static during the entire PF process. During the SIS stage, the particles can stay totally independent because the prediction and the weight update actions do not require any particle concurrency. After the SIS, the information (observations, weights and particle states) need only be exchanged between particles that are neighbours. A suitable hardware implementation of our proposal would be to treat each particle being implemented by one processing element (PE); that is, the PE would resample to select one particle from amongst its neighbors. The routing between particles is fixed throughout. The particles are routed according to the topology described in $\mathbf{G}$, i.e., if the particle $\bar{\mathbf{x}}_t^i$ has the set of $J-1$ particles $\{\bar{\mathbf{x}}_t^j\}_{j=1, j\neq i}^{J-1}$ as its neighbours, then a permanent electronic connection is established between each of the $J-1$ particles with $\bar{\mathbf{x}}_t^i$. The resampling is performed as follows.

Local Stochastic resampling: We compute the neighbor set for each particle as

$$S_i = \{s \ : \ \mathbf{G}_{i,j=1:N} = 1\}, \qquad i = 1, \ldots, N \tag{11}$$

The cardinality of the set $|S_i| = J \ \forall i$. We then generate the set of i.i.d. random samples $u^i \sim \mathcal{U}(0,1), i = 1, \ldots, N$ and then

Algorithm 1: Local stochastic network resampling $\{\mathbf{x}_t^i, w_t^i\}_{i=1}^N = \text{Resample}\left[\{\bar{\mathbf{x}}_t^i, \bar{w}_t^i\}_{i=1}^N, \mathbf{y}_t, S\right]$.

for $i = 1, \ldots, N$ in parallel **do**
  Generate the random sample $u^i \sim \mathcal{U}(0, 1]$
  Obtain the neighbor weights for the ith particle:
    $a_t^{1:J} \leftarrow \{\bar{w}_t^{S_i(j)}, j = 1, \ldots, J\}$
  for $j = 1, \ldots, J$ **do**
    Normalise the weights $a_t^j \leftarrow a_t^j / \sum_{s=1}^J a_t^s$
  end for
  Initialise CDF $c^1 \leftarrow 0$
  for $j = 2, \ldots, J$ **do**
    Construct CDF $c^j \leftarrow c^{j-1} + a_t^j$
  end for
  Set $j \leftarrow 1$
  while $u^i > c^j$ **do**
    $j \leftarrow j + 1$
  end while
  Set $\mathbf{x}_t^i \leftarrow \bar{\mathbf{x}}_t^{S_i(j)}$ and $w_t^i \leftarrow 1/N$
end for

set the index

$$I(i) = S_i(j) \ : \ u_i \in \left(\sum_{s=S_i(1)}^{S_i(j-1)} \bar{w}_t^s, \sum_{s=S_i(1)}^{S_i(j)} \bar{w}_t^s \right) \tag{12}$$

for $i = 1, \ldots, N$. Once the resampling indices $I(i), i = 1, \ldots, N$ are computed, the final weighted particle set can be obtained according to (10). We call this procedure the "local stochastic network resampling variant" because we perform random resampling locally within the neighbors of each particle. The algorithm for this stochastic variant is shown in Algorithm 1. Here a particle $\bar{\mathbf{x}}_t^i$ and its $J-1$ neighbours can be resampled stochastically in accordance to their weights to draw one particle in a non-greedy fashion by comparison of the cumulative weight sum with a single draw from $\mathcal{U}(0, 1]$. The output weights are then reset to $1/N$. This procedure will ensure that small weights are not neglected all the time, but comes at the expense of extra interaction between the particles to allow normalisation. The inter-particle routing, however, is now limited to only J particles that are still connected based on a fixed topology. The lower the value of J, the more sparse is the inter-particle dependence. This in turn minimizes the data exchange among particles and accelerates the PF operation.

Local greedy resampling: A different approach is to set the index according to

$$I(i) = \arg\max_j \ \bar{w}_t^{S_i(j)}, \qquad j = 1, \ldots, J \tag{13}$$

where $\bar{w}_t^j$ in this case are the unnormalised weights. Again, once the resampling indices $I(i), i = 1, \ldots, N$ are computed, the final weighted particle set can be obtained according to (10). This is a purely deterministic approach, which we term the "local greedy network resampling variant" because the approach selects the particle with the maximum weight from amongst the neighbors. The algorithm for this stochastic variant is shown in Algorithm 2. Here a particle $\bar{\mathbf{x}}_t^i$ and its $J-1$ neighbours are

Algorithm 2: Local greedy network resampling $\{\mathbf{x}_t^i, w_t^i\}_{i=1}^N = \text{Resample}\left[\{\bar{\mathbf{x}}_t^i, \bar{w}_t^i\}_{i=1}^N, \mathbf{y}_t, S\right]$.

for $i = 1, \ldots, N$ in parallel **do**
  Generate the random sample $u^i \sim \mathcal{U}(0, 1]$
  Obtain the neighbor weights of the ith particle:
    $a_t^{1:J} \leftarrow \{\bar{w}_t^{S_i(j)}, j = 1, \ldots, J\}$
  Reorder $S_i(j = 1 : J)$ such that $a_t^{j=1} \leq \cdots, \leq a_t^{j=J}$
  Set $\mathbf{x}_t^i \leftarrow \bar{\mathbf{x}}_t^{S_i(J)}$ and $w_t^i \leftarrow 1/N$
end for

deterministically resampled to draw a single particle by choosing the particle with the maximum weight from the list and setting its weight to $1/N$. This process is repeated for all N particles. Essentially, each particle is equipped with a comparator that chooses the particle with the maximal weight from its J entries. The key advantages of this scheme are that we only need to select the maximal weight particle from amongst J neighbors, and hence we fully avoid the need to normalize weights and the process of searching along the cumulative sum of weights.

Although this technique can be safely used in low noise conditions, e.g., aircraft tracking, its performance will deteriorate in high noise scenarios. This is because the procedure is greedy and resamples only the best particles, i.e., the lower weight particles are mostly ignored. Then the PF in effect becomes a mode tracker [40] rather than a density estimator. That is to say, greedily selecting the best local particle significantly accelerates the PF operation, but at the expense of losing diversity among the particles, for example, in losing potential information when the target is covert or makes a sharp maneuver [22]. This information loss is reduced when the stochastic resampling variant is used.

The key benefit of the proposed network resampling is that the length of the climb along the cumulative sum ladder for each i is limited to a maximum of J steps, with $J \lll N$ even for millions of particles. Moreover although the climb is sequential, the entire operation over $i \in \{1, \ldots, N\}$ is fully parallel thus rendering the method computationally friendly. The maximum order of complexity for one time step for this proposed resampling will be $\mathcal{O}(NJ)$. We assume here that the observations are globally distributed in a synchronous manner to all the particles of a network, but the PF could perhaps be constructed so that the observations are propagated to each particles' neighbours.

The proposed fixed network based resampling is diagramatically illustrated using a simple example in Fig. 2.

B. Discussion

Here a brief comparison of the merits of the proposed over the state-of-the-art is presented. The interactions for the distributed computing PF [16] are shown in Fig. 3(a). Here, all the $N = 256$ particles are assigned to $M = 8$ local filters, each containing 32 particles. Each local filter (the white portion) is assigned to a PE. The PE resamples the particles in the local filter and $\chi \geq 1$ particles with large weights from each of its neighbours (the gray portions). Here, the synchronization requirements and the number of inter-particle interactions for each particle is limited

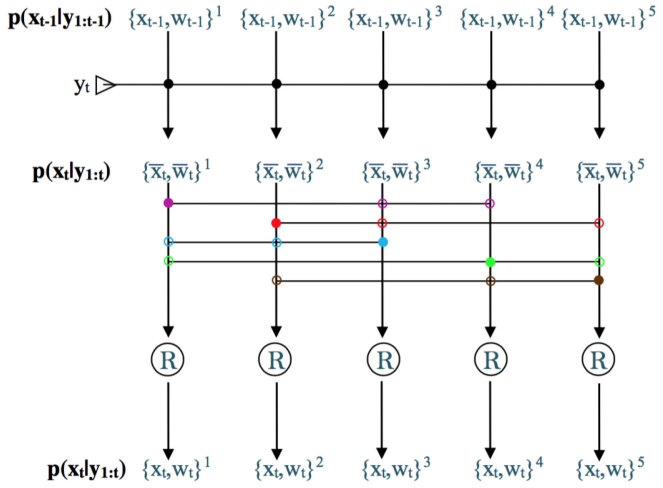
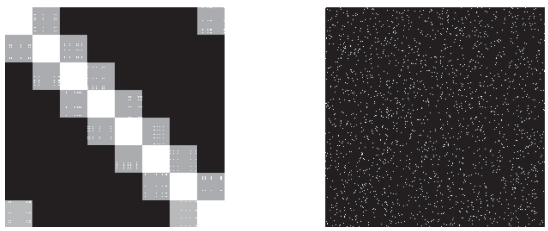


Fig. 2. The use of a network topology as a fixed resampling unit in the PF for $N = 5$ and $J = 3$. Initially there are 5 weighted particles representing the posterior $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ at time t. At the next time step t, each particle is loaded with the sensor data $\mathbf{y}_t$. In this example, the observation $\mathbf{y}_t$ is received synchronously by all nodes, although that is not an essential feature of the proposed system. Next, a new set of particles is sampled and the weights are updated. The particles are then routed according to the $\mathbf{G}$ obtained using any of the networks described in section III. It can be observed that in the first row $\bar{\mathbf{x}}_t^1 \rightarrow \{\bar{\mathbf{x}}_t^3, \bar{\mathbf{x}}_t^4\}$ (the source node is denoted by a solid dot and its neighbours by circles), in the second row $\bar{\mathbf{x}}_t^2 \rightarrow \{\bar{\mathbf{x}}_t^3, \bar{\mathbf{x}}_t^5\}$ and so on. In the last stage, each particle and its neighbours are fed to the resampler ® that examines the J incoming weights and outputs the chosen particle. The final particle set that approximates $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ is thus formed. ® is a comparator for the greedy resampling variant and a multinomial resampling unit for the stochastic resampling variant.



(a) Distributed computing PF. (b) Metropolis.

Fig. 3. The particle interactions in state-of-the-art resamplers. The number of particles is $N = 256$. In (a), the number of local filters is $M = 8$ and the number of neighbour interactions is $2\chi = 20$. In (b), each particle compares with $B = \log_2(N) = 8$ other particles.

to $N/M + 2\chi$. The interaction relation for the Metropolis resampler [24], [26] at a single time sample is shown in Fig. 3(b). Here the number of particles each particle is compared with is chosen to be $B = \log_2(N) = 8$. The interaction is limited to only $B = 8$ particles with no need of normalization. The conventional systematic resamplers [2], [27] include weight normalization, and the interaction graph of these resamplers is fully connected leading to exhaustive communication within the particles. In all these methods, the set of particles each particle interacts with are unknown beforehand. Consequently, the electronic routing between the particles is volatile and this causes substantial delay in processing the incoming sensor data. There is no previous resampler that does not require changing the inter-particle communication architecture for resampling every time other than the proposed fixed network resampler.

The inter-particle interaction for the proposed fixed network resampler (greedy and stochastic variants) can be understood from the adjacency matrix $\mathbf{G}$ of a small-world network shown in Fig. 1 for $J = \log_2(N)$. Here every particle needs to interact with only $J - 1$ other particles every time sample. This resampling structure is computed only once and remains fixed throughout the PF procedure. Moreover, the proposed technique requires only local normalization (or none), and hence does not require synchronization of all particles. Therefore, we claim that our proposed method is highly suitable for PF implementation in parallel hardware architectures where conventional resamplers fail, e.g., even for a million particles with $J = \log_2 N$, each particle interacts with only 20 other particles. The use of fixed network architecture for PF resampling was first proposed in 2014 [41]. This paper now generalizes the idea in a broader context of using any sparsely connected network that has full information exchange for PF resampling, and establishes the properties and the validity of the method for tracking in high dimensions using a very large number of particles. The key merits of the proposed method over the state-of-the-art are, (a) a substantial reduction in the communication overhead induced in conventional resampling, and (b) non-volatile resampling, i.e., the routing of particles is fixed throughout the filtering process, thus rendering the method extremely parallel friendly.

C. Choice of the Network

The choice of the network is critical to achieve computational feasibility. A good choice for a network is one that has (a) complete information transfer from any node to every other node (i.e., there are no islands of nodes) and, (b) minimal connectivity between the nodes. The sparse small-world networks are clearly an ideal choice to satisfy these criteria. Some others, like the ring, the scale-free or the hub networks do not satisfy the aforementioned criteria and hence are not suitable for the proposed PF resampling.

V. Evaluation and Discussion

In this section, the efficacy of the proposed fixed network based resampler is analysed. We first check that the proposed method is truly valid and works for the general class of state space models. For this we analyse the bias properties of the resampler and show that our proposal, like most conventional resamplers, exhibits unbiasedness in its resampling of weighted particles. Secondly, we test our method on a linear Gaussian system. This model is chosen to evaluate the performance of the proposed resampling method in terms of its ability to accurately represent the posterior given by the optimal Kalman filter. We also discuss how the proposed fixed network resampling affects particle clusters and weights. For the same model, the ability of the proposed method to effectively track extended targets in the presence of false alarms and missed targets observations is also examined. The true efficacy of the proposed method becomes evident when employing the PF in high dimensional nonlinear models. So as a final testing scenario, we employ a nonlinear

motion model with range and bearing tracking of five targets jointly. This renders the state space and the observation space high dimensional and requires a large number of particles for accurate tracking. For this we show that where the conventional resamplers fail, our proposed method can be used effectively.

The PF operating on the proposed fixed network based resampler is compared with the Kalman filter [42] (in the linear Gaussian case), and PFs operating with the stochastic [2], [18]–[21], [23], deterministic [9], [11], [22], Metropolis [24], coalesced Metropolis [26], distributed PF [16] and parallel systematic/stratified [27] resamplers. For ease of visualization, not all the possible existing methods are shown. It was found that the performance of the multinomial, the stratified, the residual and the residual systematic resamplers was almost identical to that of the popular systematic resampler and hence only results for the latter are shown. Moreover, the proportional redistribution [11], the partial deterministic and the soft resamplers exhibit similar performance and hence only the last of these is shown. This applies to all the results presented in this section. It was also found that the choice of the network structures described in section III made little difference for the proposed technique because the rate of transfer of information within the particles is similar for the chosen value of J. Hence for further ease of visualization, only the FDR network based resampling is shown for comparison with the various existing resampling schemes. The PF performance using the different graph structures is then analysed separately. The performance of the proposed method at different degrees of connectivity J is also presented. A key difference in this article from the first that proposed the use of fixed network architectures for PF resampling [41] is that this article presents an evaluation study of the properties of the proposed technique in high dimensions.

All the results in the sequel are averaged over 100 simulations.

A. Bias Analysis

In this section, we analyse the proposed resampler on the basis of two bias properties, (a) proper weighting, and (b) likelihood values. Comparing (5) and (6) indicates that the number of replications of each particle in order to obtain an unbiased approximation to the posterior should be $n_t^i = N\bar{w}_t^i, i = 1, \ldots, N$. This is called the proper weighting condition [4], [5]. A reliable measure of the resampling quality in this context is the mean of the distance between the integral approximations (5) and (6). This mean distance [5] is given by

$$\mathbb{E}\left[\sum_{i=1}^{N} \frac{n_t^i - N\bar{w}_t^i}{N}\delta(\mathbf{x}_t)\right]. \tag{14}$$

Restricting the number of replications n_t^i to lie close to $N\bar{w}_t^i$ for $i = 1, \ldots, N$ reduces the variance and hence satisfies the proper weighting criterion. For testing this property, we chose the weights to be $\bar{w}^i = \exp(si), i = 1, \ldots, N$ where $s \in (0.03, 0.3)$ is the scale parameter. The number of samples is chosen to be $N = 128$. A low value of $s = 0.03$ creates many particles having moderate weights and a large value of s creates particles of more widely varying weights. We test many cases between these two extreme scenarios.

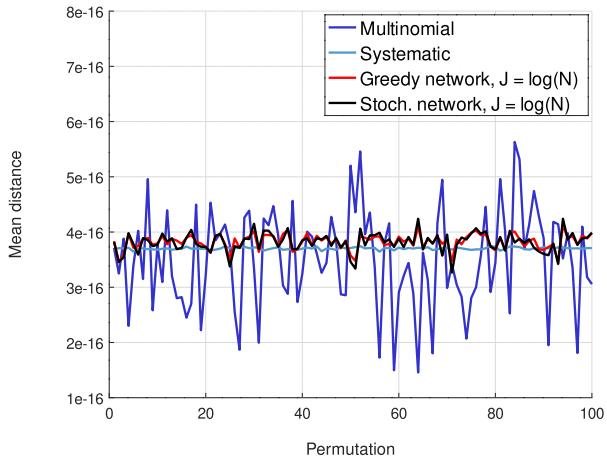


Fig. 4. The mean distance between the integral approximations in (5) and (6) is zero to within machine precision for all the resamplers tried for 100 permutations of the weights for scale $s = 0.2$.

The mean distance of the standard multinomial and systematic resamplers and the proposed network resamplers is zero to within machine precision and similar for all scale parameters. The scale dependence is not shown. Fig. 4 shows the mean distance between the integral approximations in (5) and (6) at $s = 0.2$, for 100 permutations of the weights. This test helps understand the efficacy of information transfer from one sample to every other sample when the samples are minimally connected and the dominating weight could be from any sample. It can be seen that the distance is effectively zero, indicating that the proposed resampler indeed replicates particles with a close-to-proper weighting condition. Moreover the proposed resamplers maintain consistency in the mean distance over many permutations of the weights indicating that the minimal connection does not adversely affect the propagation of information from one particle to every other particle.

Secondly, a direct consequence of an unbiased approximation of the posterior is high resampling likelihood value. The likelihood function for multinomial selection is given by [5]

$$P(N^1 = n^1, \ldots, N^n = n^n) = \frac{N!}{n^1! \ldots n^n!}(\bar{w}^1)^{n^1} \ldots (\bar{w}^n)^{n^n} \tag{15}$$

where the total number of replications of the particles satisfy $\sum_{i=1}^{N} n^i = N$. This function is maximized at $n^i = Nw^i$ and since the resamplers have their replications n^i close to Nw^i as shown in Fig. 4, their average likelihood is also high. Fig. 5 shows the log likelihood of the proposed resamplers using FDR networks and the conventional multinomial and systematic resamplers. The multinomial resampling has the lowest likelihood due to high Monte Carlo error in its estimates and the systematic exhibits the highest likelihood values by virtue of reduced Monte Carlo error variance [5]. It can be observed that the proposed fixed network resampler has higher likelihood values than the multinomial and close to that of the systematic resampler. Towards $s = 0.3$, the network resamplers take extra time to propagate the information contained in the few large

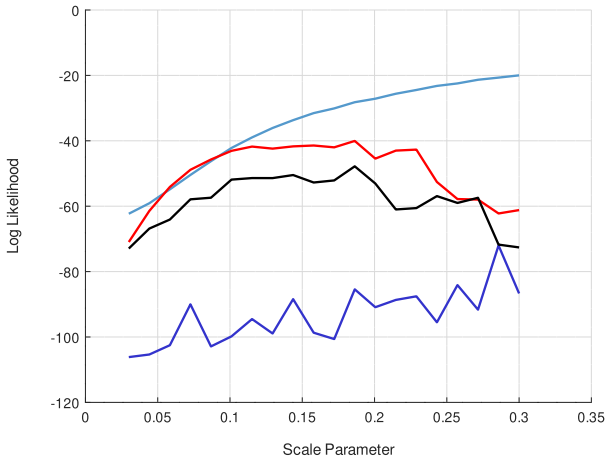


Fig. 5. The log-likelihood for the resamplers versus the scale parameter. The legend of Fig. 4 applies to this figure.

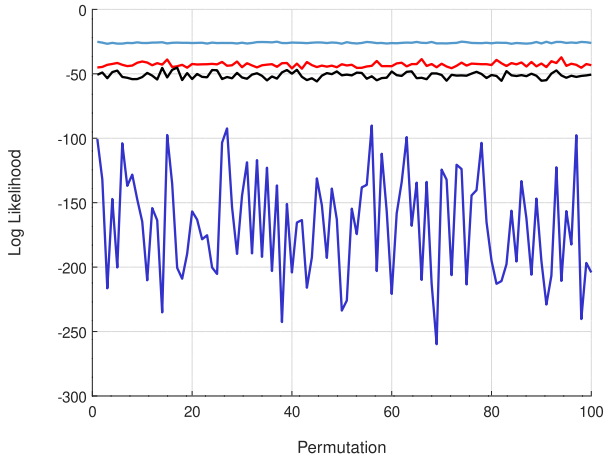


Fig. 6. The log-likelihood for the resamplers for 100 permutations of the same weight sequence with $s = 0.2$. The legend of Fig. 4 applies to this figure.

weight particles to the many low weight particles and this causes a decrease in the likelihood values. However, it can be seen that the likelihood values of the proposed network resampler falls within the performance range of the conventional resamplers.

Fig. 6 shows the log-likelihood for the weight sequence with $s = 0.2$ for 100 permutations of the same sequence. It can be observed that the proposed resamplers maintain consistency over all the 100 permutations. This again indicates that the proposed minimal interaction resampling method does not adversely affect the information transfer within samples. In contrast, it can be understood that the proposed minimally interacting parallel-friendly resampling method incorporates a communication overhead that is just sufficient for accurate and valid resampling. These two tests confirm that the proposed fixed network resampling technique conforms to the desired properties of a resampler in giving an unbiased estimate of the posterior.

B. Testing Using Linear Gaussian Model

The proposed method is tested for tracking targets which are point mass representations in the state space, i.e., a target generates a single point-measurement. For this, a six-dimensional linear Gaussian constant acceleration (CA) state transition model is used so that comparison can be made with the Kalman filter [42] which is known to give the optimal posterior. The target is a six dimensional vector $\mathbf{x}_t = (x, v_x, a_x, y, v_y, a_y)^\top$ where (x, y) are the positions and $v_{(.)}$, $a_{(.)}$ are the corresponding velocity and acceleration components. The process model is

$$\mathbf{x}_t = \begin{bmatrix} 1 & \delta t & \delta t^2/2 & 0 & 0 & 0 \\ 0 & 1 & \delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \delta t & \delta t^2/2 \\ 0 & 0 & 0 & 0 & 1 & \delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_{t-1} + \mathbf{a}_t \qquad (16)$$

for $t = 1, \ldots, T = 100$, where $\mathbf{a}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$ with the process covariance matrix $\mathbf{Q} = \text{diag}[4, 4, 0.1, 4, 4, 0.1]$.

The sensor records a clean noisy point measurement of the target position. Here, the term "clean" implies that despite added noise, there are no false alarms, clutter detection, missed observations and/or out-of-sequence (OOS) observations. The sensor model is

$$\mathbf{y}_t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \mathbf{x}_t + \mathbf{e}_t \qquad (17)$$

where the sensor noise is $\mathbf{e}_t \sim \mathcal{N}(0, \mathbf{R} = \sigma^2 \mathbf{I}_2)$ where $\sigma^2 = 5$. The initial target state is $\mathbf{x}_{t=0} = [0, 0, 0, 0, 0, 0]^\top$. The PFs are initialized with $p(\mathbf{x}_t) = \mathcal{N}(\mathbf{0}, \mathbf{I}_6)$.

1) Kolmogorov-Smirnov Statistic: Firstly, the faithfulness of the proposed fixed network based resampler in accurately representing the posterior is tested using the well known Kolmogorov-Smirnov (KS) statistic [43]. It is known that for linear Gaussian systems, the Kalman filter [42] provides a optimal tracking estimate. This paper proposes to test the faithfulness of the PF in accordance to its KS statistic disagreement with the *theoretically optimal* Kalman filter. The KS statistic is known to provide a reliable measure of the accuracy of the estimate of the posterior. The procedure is as follows: After resampling, the particles are de-meaned and de-correlated using the theoretically optimal Kalman mean and covariance as

$$\mathbf{m}_t^i = (\mathbf{S}^{\text{KF}})^{-1/2}(\mathbf{x}_t^i - \hat{\mathbf{x}}_t^{\text{KF}}), \quad i = 1, \ldots, N \qquad (18)$$

where $\hat{\mathbf{x}}_t^{\text{KF}}, \mathbf{S}^{\text{KF}}$ are the estimated mean and covariance of the Kalman filter at time t. This would result in a set of uncorrelated particles with zero mean and unit variance if the PF distribution were exactly identical to that of the Kalman filter. Then the particles along each of the jth dimension are sorted as $\{w_t^i, \mathbf{m}_t^i(j)\}_{i=1}^N : \mathbf{m}_t^i(j) \leq \mathbf{m}_t^{i+1}(j),\ i = 1, \ldots, N$ and the cumulative sum of the corresponding weights and the error function of the de-meaned samples are computed according to

$$c_t^i = \sum_{a=1}^{N} w_t^a, \qquad i = 1, \ldots, N \qquad (19)$$

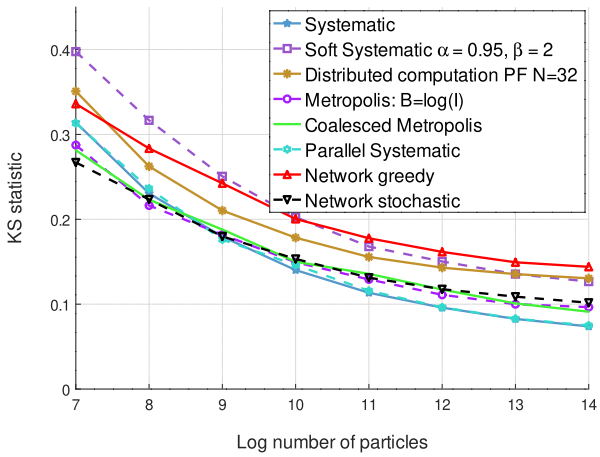


Fig. 7. The KS statistic versus the number of particles N. The networks are formed with connectivity degree $J = \log_2(N)$.

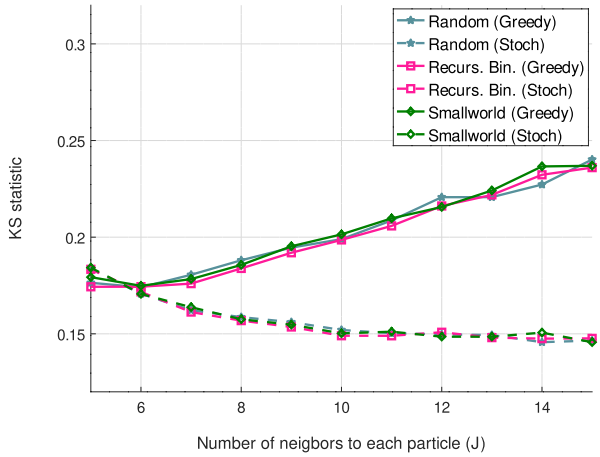


Fig. 8. The KS statistic versus the number of the neighbors J of each particle for the aforementioned graph structures.

$$e_t^i = \frac{1 + \text{erf}(\mathbf{m}_t^i(j)/\sqrt{2})}{2}, \quad i = 1, \ldots, N \tag{20}$$

Finally, the one-sample KS test is conducted along each dimension separately (by comparing with the CDF of a zero mean unit variance normal distribution) and the maximum KS deviation value amongst all the dimensions is taken. This is expressed as

$$\kappa_t = \max\left\{\max\left\{|c_t^{i=1:N} - e_t^{i=1:N}|\right\},\ j = 1, \ldots, D_{\mathbf{x}}\right\} \tag{21}$$

Fig. 7 shows the KS statistic of the PFs operating with different resamplers for varying numbers of particles. It can be inferred from the figure that the greater the number of particles, the better is the agreement between the PF and the Kalman filter [3]. It can be observed that while the resource-hungry and parallel-unfriendly systematic resampler exhibits the best performance, the proposed parallel-friendly and minimally interacting fixed network based resampler can achieve the same performance by using extra particles. Although the use of more particles increases the computational load, this computation may be more achievable in the proposed device since parallelization is now possible. For example, the performance obtained by the systematic resampler at 256 and 1024 particles can be achieved by using the proposed resampler at nearly 1024 and 16 384 particles respectively in the greedy variant, and the stochastic variant performs equivalently. The stochastic variant is more effective than the greedy variant by virtue of leveraging the lower weight particles in the resampling process. It is observed that the performance of all the resamplers converges on the optimal for increasing numbers of particles N as theoretically guaranteed in [44]. With regard to the KS statistic of the PFs operating on the network topologies described in section III, It has been observed that all the networks approximate the posterior almost identically. Hence their KS statistics are not shown individually.

Fig. 8 shows the KS statistic of the greedy and stochastic versions of network sampling at varying degrees of connectivity J. It can be seen that the performance of the stochastic variant improves with increasing J. The performance of the greedy variant improves with decreasing J and low values of J aid in better PF acceleration and easier parallelization.

At $J \leq 5$, the greedy variant selects the maximum weight among fewer particles leading to sustainability of moderate weights while the stochastic variant selects randomly among fewer particles resulting in high probability of losing large weights. At $J > 5$, the greedy variant selects the maximum weight among many particles leading to loss of moderate weights, i.e., the network is too strongly connected to be able to sustain more information, while the stochastic variant selects randomly among many particles thus leading to survival of moderate weights. In the greedy variant with strong connectivity, the largest weight particle very quickly propagates through the network thus leading to loss of diversity. By contrast, the stochastic variant with strong connectivity leads to improved performance because it makes the resampler behave more like the classical resamplers. Therefore the stochastic variant of fixed network resampling approximates the posterior more accurately for incrementing degree of connectivity J.

2) Effect of Proposed Resampling on Degeneracy: The purpose of resampling is to mitigate the effect of sample degeneracy. Resampling leads to, (a) reduction in the variance of the weights, and (b) tightness in the sample cluster. A measure of the variance in the weights is the estimated effective sample size $\hat{N}_{\text{eff}} = 1/(\sum_{i=1}^{I} w_t^{i\,2})$ where $\hat{N}_{\text{eff}} \in (0, N)$. The effect of degeneracy and the consequential effect of resampling using the proposed network resampler is similar to the case of using the traditional resamplers (and hence the figure is not shown here). The effect of resampling on the sampler cluster can be observed in the particle cluster tightness, which is defined as

$$c_t = \exp\left(-1/N \sum_{i=1}^{N} ||\mathbf{x}_t^i - \hat{\mathbf{x}}_t||^2\right) \tag{22}$$

Fig. 9 shows the value of $\log(c_t)$. The systematic and parallel systematic resampler show similar performance so the former is shown. The soft systematic, metropolis, coalesced metropolis, network greedy and network stochastic resamplers show

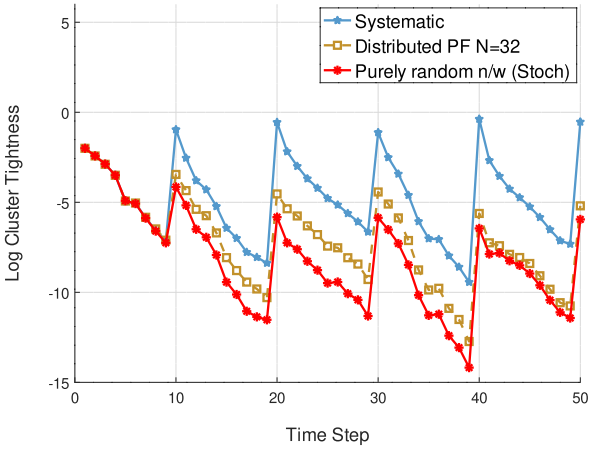


Fig. 9. The cluster tightness versus the time step. Resampling is performed once every ten time steps. The number of particles is $N = 1024$. For the network based resampler, we set $J = \log_2(N)$.

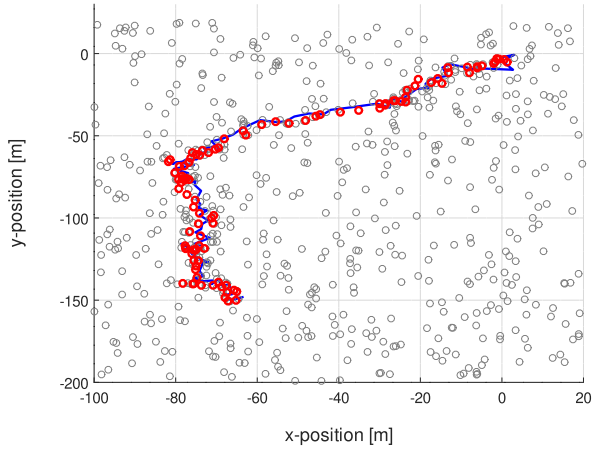


Fig. 10. A filter tracking illustration for the stochastic variant of the proposed fixed small-world network resampler for one realisation for extended target tracking in the presence of false alarms and missed detections. The blue solid line is the ground truth, the grey circles are the measurements, and the red circles are the filter estimates. The number of particles is $N = 512$ and $J = \log_2(N)$. The detection probability is $p_D = 0.8$. Other specifications are as given in the text. The greedy variant shows similar performance.

similar performance so only the latter is shown. The network and Metropolis resamplers with minimal particle interaction take more time to gain the tightness of equivalent to that of the systematic resampler that involves exhaustive inter-particle interaction.

3) Extended Target Tracking With False Alarms and Missed Detections: Now, the ability of the proposed method to track an extended target in the presence of false alarms and missed observations is tested. An extended target in this context is one that generates one or more point target detections per target. This testing scenario also illustrates an advantage of relatively slow information transfer in sparse networks. The process model described in (16) is the same. The observation model in (17) will now be

$$\mathbf{z}_t = \{\mathbf{y}_t^i\}_{i=1}^{E_t} \cup \{\mathbf{f}_t^j\}_{j=1}^{F_t} \tag{23}$$

where the number of observations per target is $E_t \sim \mathcal{U}(0, E_{\max} = 2)$ and the number of false alarms are $F_t \sim \text{Poisson}(\lambda_F = 5)$. The false alarms are detected uniformly from the observation space as $f_t(x)^i \sim \mathcal{U}(x_{\min}, x_{\max})$ and $f_t(y)^i \sim \mathcal{U}(y_{\min}, y_{\max})$. The target observations $\{\mathbf{y}_t^i\}_{i=1}^{E_t}$ are detected with detection probability p_{D}.

Fig. 10 illustrates the PF planar estimates for the same realisation for $N = 2^8$ of the proposed fixed network resampler. It can be seen that the proposed method tracks with good accuracy. A simulation video for this scenario is available.[1]

For the system mentioned above, Fig. 11 shows the dependence of the rmse on $1 - p_{\text{D}}$. Fig. 12 shows a close-up of Fig. 11 for the proposed fixed network and Metropolis resamplers. When the target observations are not available for some duration (or when the sensor detects false alarms close to the target position), the particle weights gradually decrease and the particles start to lose the target and diverge. Once several correct observations have arrived (or the false alarms in the vicinity of the particles have disappeared), the weights of

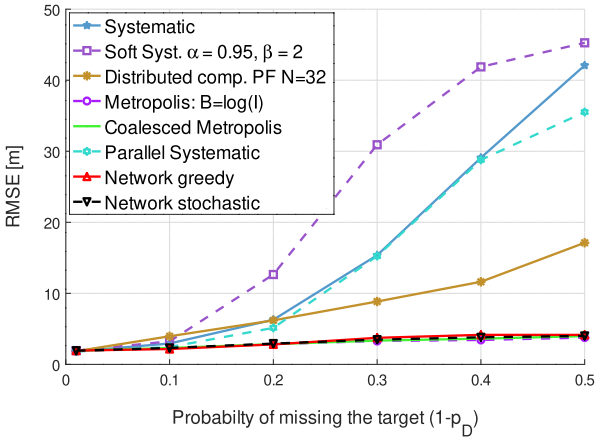


Fig. 11. The RMSE versus the probability that a sensor misses an observation. The number of particles $N = 512$.

the particles near the target start to gradually increase. Unlike the deterministic resamplers, the stochastic resamplers do not have any memory of the goodness of a particle, i.e., all particles are assumed to be equally valid in tracking the posterior. Hence they sense the change in the particle weights faster and converge rapidly towards the true target location. The low estimation error of the stochastic resamplers indicates this. In contrast, the deterministic resamplers retain the information contained in the particle weights [22] and hence require more time to change their belief about a particle. It can also be observed in the figure that the proposed fixed random network based resampler and the Metropolis resamplers exhibit higher track accuracy because the divergence of the particles from the true target location (due to the information loss caused by false alarms and missed detections) is slower than that of the other techniques. Fig. 12 again shows that the choice among of three network types makes

[1][Online]. Available: https://bit.ly/3hsBLJC

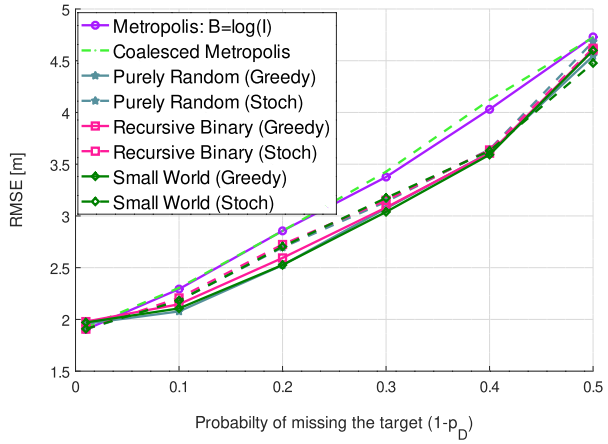


Fig. 12. The RMSE versus the probability that a sensor misses an observation. This figure shows a close-up of Fig. 11 for the proposed network and Metropolis resamplers. The legend of Fig. 11 applies to this figure also.

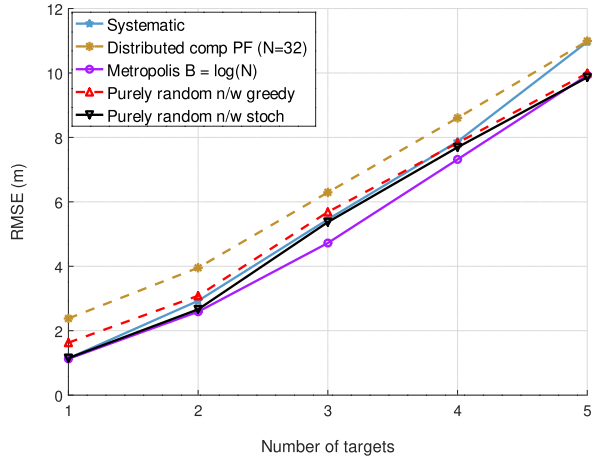


Fig. 13. The RMSE versus the number of targets with $N = 1024$ for various resamplers compared with the FDR fixed network resampler.

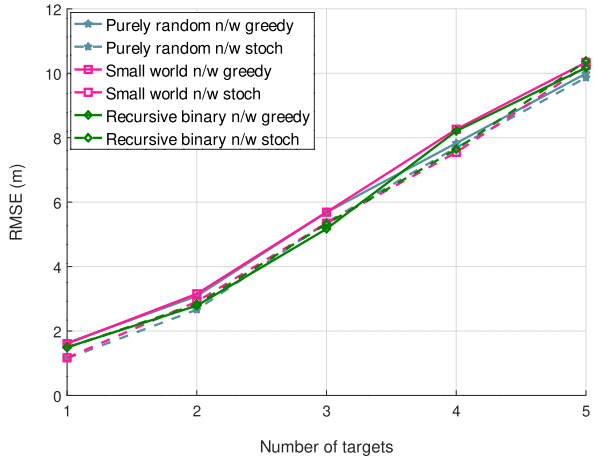


Fig. 14. The RMSE versus the number of targets with $N = 1024$ for several sparse fixed network resamplers.

little difference to the tracking accuracy. Matlab code for the proposed fixed network resampler and other resamplers tested here is available.[2]

C. High Dimensional Nonlinear Multi-Target Tracking

In this section, we demonstrate the performance of the proposed fixed network resampling method in comparison with the conventional resamplers for a high dimensional nonlinear multiple target tracking example. Each target has a four dimensional state vector $\mathbf{x}_t = (x, y, v_x, v_y)^\top$ where (x, y) are the positions and $v_{(.)}$ are the corresponding velocities. The process model follows a nonlinear constant turn (CT) model

$$\mathbf{x}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{C}\mathbf{a}_t, \quad t = 1, \ldots, T = 100 \tag{24}$$

where

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \dfrac{\sin(\omega_t \delta t)}{\omega_t} & -\dfrac{1-\cos(\omega_t \delta t)}{\omega_t} \\ 0 & 1 & \dfrac{1-\cos(\omega_t \delta t)}{\omega_t} & \dfrac{\sin(\omega_t \delta t)}{\omega_t} \\ 0 & 0 & \cos(\omega_t \delta t) & -\sin(\omega_t \delta t) \\ 0 & 0 & \sin(\omega_t \delta t) & \cos(\omega_t \delta t) \end{bmatrix},$$

$$\mathbf{C} = \begin{bmatrix} \delta t^2/2 & 0 \\ 0 & \delta t^2/2 \\ \delta t & 0 \\ 0 & \delta t^2/2 \end{bmatrix}$$

δt is the time interval between two observations and the turn rate follows a random walk model according to $\omega_t = \omega_{t-1} + \delta t u_{t-1}, u_{t-1} \sim \mathcal{N}(0, \sigma_u^2 \mathbf{I})$ with $\sigma_u^2 = \pi/180$ rad/sec. The process noise is $\mathbf{a}_t \sim \mathcal{N}(0, \tau^2 \mathbf{I})$ with $\tau^2 = 10$ m/s^2. The target is observed via its range and bearing as

$$\mathbf{y}_t = [r_t, \theta_t]^\top + \mathbf{e}_t \tag{25}$$

[2]https://bit.ly/3kwJE30. mn_linmodel.m will run the simulation of the model in (16) and (17), and mn_linmodel_efa.m will run the simulation for extended target tracking in the presence of false alarms and missed observations.

where

$$r_t = \sqrt{x^2 + y^2}, \ \theta_t = \tan^{-1}(y/x) \tag{26}$$

where the observer is at the origin (0,0) and the sensor noise is zero mean Gaussian as $\mathbf{e}_t \sim \mathcal{N}(0, \mathbf{R})$ where $\mathbf{R} = \text{diag}(\sigma_r^2, \sigma_\theta^2)$ where $\sigma_r^2 = 5$ m^2, $\sigma_\theta^2 = 0.45$ rad^2. For this example, we consider the tracking of N_T targets starting randomly in the surveillance region of $(-35, 35)$ along the x and y directions. We use the joint filtering approach where the states of the targets are concatenated as a single joint state. The turn rate is assumed to be the same for all the targets. So the joint target state for $N_T = 5$ is now $\mathbf{X}_t = (\mathbf{x}_t[1], \mathbf{x}_t[1], \ldots, \mathbf{x}_t[Nt], \omega_t)^\top$ with twenty one dimensions. The PF is initialized with particles uniformly distributed in the surveillance region. For this scenario, the PF requires many particles to explore the very high dimensional state space in order to lock onto the targets and track them accurately.

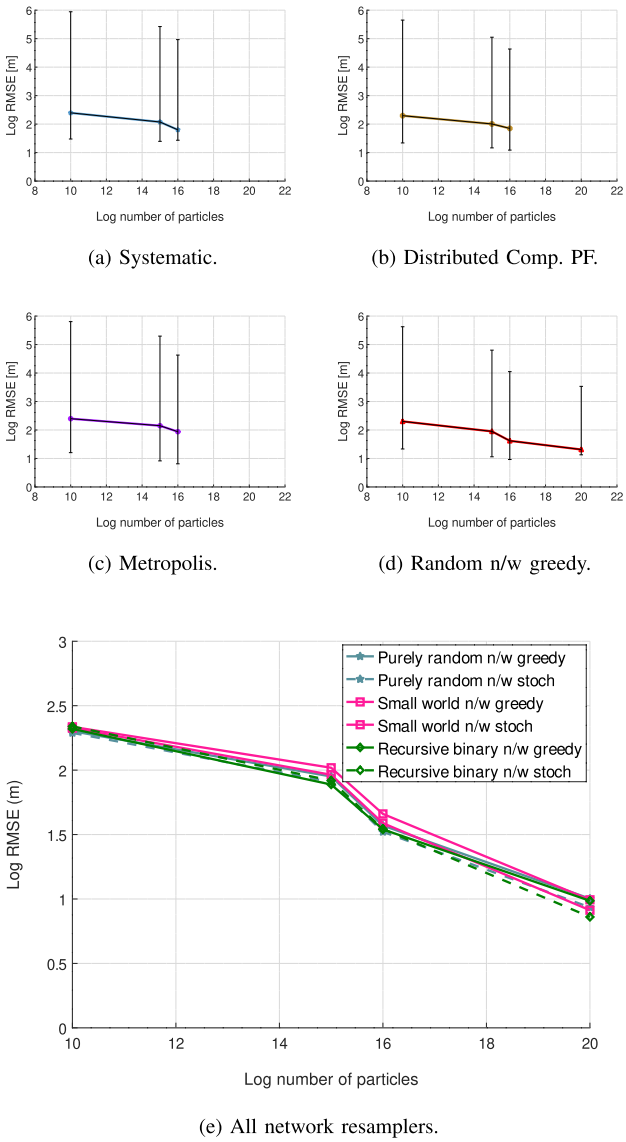
Figs. 13 and 14 show the tracking accuracy measured as RMSE between the ground truth and the filter estimate versus the

number of targets with the number of particles $N = 1\,024$. For convenience, we test the most popularly used systematic resampler, the Metropolis resampler that has lately attracted attention in the research community and the distributed computation PF and compare them with the proposed fixed network resamplers. It can be observed that the proposed fixed network resampler exhibits performance equivalently to that of the conventionally used systematic and the other parallel friendly resamplers. As the number of targets increases, the dimensionality of the state space increases. This consequently requires a large number of particles to track with low RMSE. It can be observed that the tracking efficiency of the systematic resampler reduces after $N_T = 4$ (seventeen dimensional state space). This tracking example is a classic example to show that where the conventional resamplers fail, the proposed fixed network greedy and stochastic resamplers continue to operate using a larger number of particles.

We now test for joint tracking of $N_T = 5$ (five) targets. Simulation videos of PF tracking using fixed network resamplers for two different realisations with different filter initializations for $N_T = 5$ targets using $N = 2^{15}$ particles are available.[3] It can be seen that the proposed method tracks with satisfactory accuracy. Fig. 15 shows log RMSE with 95% confidence bars versus the log number of particles for $N_T = 5$. The 95% confidence ranges are shown for the conventional resamplers and the proposed network greedy variant. Since all the network resamplers exhibit similar performance, their confidence intervals are not shown. The dimensionality of the state space is 21 and that of the measurement space is 10. The RMSE is first averaged across all the realisations and then across the dimensions. It can be observed that the proposed method slightly outperforms the systematic resampler at this high dimensional operation as was evident in Fig. 13. A possible reason for this could be the non-exhaustive information transfer between particles that facilitates them to change belief regarding the target rather slowly, thus giving leverage to explore the state space longer before locking on the target; this aids in avoidance of a possible filter divergence.

The use of more particles improves the approximation accuracy and consequently the tracking accuracy. We test the methods for $N = 2^{10} = 1024, 2^{15} = 32\,768$ and $N = 2^{16} = 65\,536$. The key feature of the proposed method is that it facilitates the use of a large number of particles by virtue of substantially reduced communication within the particles. Thus our method is feasible using over a million particles. For $N = 2^{20} = 1\,048\,576$ particles it can be seen in the RMSE that the tracking accuracy (which is consequent on the accurate approximation of the true posterior) is lower than that at $N = 2^{16}$. The number of connections to each particle is only $J = 20$ for $N = 1\,048\,576$ particles, i.e., each particle interacts with only 20 other particles. This is a substantial reduction communication overhead between the particles over the state-of-the-art. The conventional systematic resampler becomes infeasible after some large value of N. The Metropolis resampler became highly unscalable at high $N > 65\,536$ for implementation on our computer systems due to the inclusion of the random selection of the value of B for each particle at every time step; at $N = 1\,048\,576$, the Metropolis resampler consumed nearly 24 hours to complete one $T = 100$ time step simulation, and thereby it is infeasible to run all 100 realisations, so the results are not shown. The distributed computation PF is sensitive to the choice of the number of threads and the number of particles assigned to each thread, whereas the proposed fixed network resampler is robust and a choice of $J = \log_2(N)$ is a natural choice for parallel friendly implementation without any adverse effect on the tracking performance. The PF is known to suffer from the curse of dimensionality at higher dimensions and the tracking community has always evinced interest in the real-time implementation of the PF with more than a million particles to mitigate the effect of the curse [45]. This proposed resampler has indeed facilitated the use of more than a million particles and where

[3][Online]. Available: https://bit.ly/3wBd8iB and https://bit.ly/3xsbDEu



(a) Systematic. (b) Distributed Comp. PF.

(c) Metropolis. (d) Random n/w greedy.

(e) All network resamplers.

Fig. 15. The log-RMSE versus the log number of particles for several resamplers. The black lines are the 95% confidence bars. Since the performances of the proposed network resamplers are visually identical, the confidence bars of only the random network greedy variant are shown. The RMSE of the others is shown in (e).

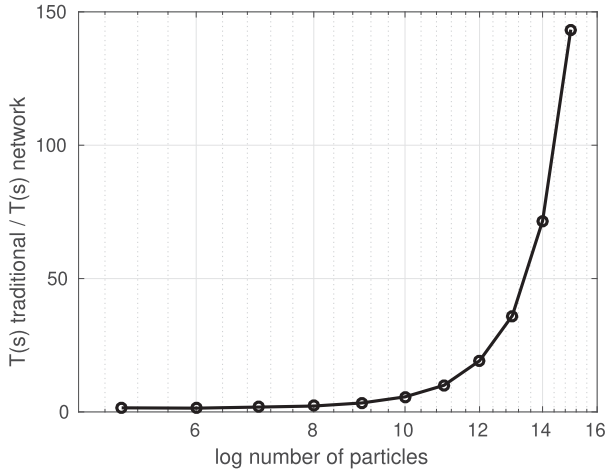


Fig. 16. The ratio of the computational time (in seconds) of the traditional method to the proposed method versus the log number of particles.

the conventional resamplers fail, the proposal works. Real-time GPU implementation will be performed in the future.

D. Computational Gain

We finally demonstrate the superiority of the proposed resampling approach in terms of computational time.Fig. 16 shows the ratio of the computation time of the particle filter using systematic resampling to that of the particle filter using the proposed random network resampling method versus the log of the number of particles for a simple tracking application implemented on a core-i5 workstation on Matlab using 4 parallel cores. It can be observed that the computational merit of our method becomes superior exponentially with the number of particles. At $2^{15} = 32\,768$ particles our method is 150 times faster than the traditional approach, and at a million particles the proposed method will be nearly $4\,800$ times faster than the systematic resampler. This is because each particle interacts with only $J = 20$ particles. Note that the speed gain available from *true* parallelization of the computation would be much greater than is shown in Fig. 15, thus method provides an enormous computational gain over state-of-the-art resampling methods

VI. Conclusion

This paper addressed the problem of PF parallelization and has presented a novel outlook for PF resampling — that communication among particles is like communication within nodes in a network. The principle of sparse networks to achieve complete information transfer with minimal communication within the nodes is employed within the PF context to substantially accelerate the PF and make it parallel friendly. The key innovation here is to use a network having a fixed architecture for PF resampling. When employed in the PF, this network provides each particle with a fixed set of a very few other particles with which it will interact and the resampler samples one particle from the set either greedily or stochastically. The proposed method was tested for bias and its tracking efficacy was evaluated in challenging high dimensional nonlinear scenarios using over a million particles. In the future, we will create a GPU implementation of the proposal and study its speed and convergence. We also aim to investigate the possibility of using adaptive and learning algorithms to automate the choice of J.

References

[1] N. J. Gordon, D. J. Salmond, and A. F. M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation," *Proc. IEE Radar Signal Process.*, Jul. 1993, vol. 140, no. 2, pp. 107–113.
[2] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, Feb. 2002.
[3] A. Doucet, N. De Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*, vol. 1. Berlin, Germany: Springer, 2001.
[4] R. Douc and O. Cappé, "Comparison of resampling schemes for particle filtering," in *Proc. ISPA 4th Int. Symp. Image Signal Process. Anal.*, 2005, pp. 64–69.
[5] J. D. Hol, T. B. Schön, and F. Gustafsson, "On resampling algorithms for particle filters," in *Proc. Nonlinear Statist. Signal Process. Workshop*, 2006, pp. 79–82.
[6] M. R. Frean, P. B. Rainy, and A. Traulsen, "The effect of population structure on the rate of evolution," in *Proc. Roy. Soc., Biol. Sci*., 2013.
[7] M. Green and M. Ohlsson, "Comparison of standard resampling methods for performance estimation of artificial neural network ensembles," in *Proc. 3rd Intl. Conf. Comput. Intell. Med. Healthcare*, 2007, pp. 25–27.
[8] I. Knollová, M. Chytrỳ, L. Tichỳ, and O. Hájek, "Stratified resampling of phytosociological databases: Some strategies for obtaining more representative data sets for classification studies," *J. Vegetation Sci.*, vol. 16, no. 4, pp. 479–486, 2005.
[9] M. Bolić, P. M. Djurić, and S. Hong, "Resampling algorithms and architectures for distributed particle filters," *IEEE Trans. Signal Process.*, vol. 53, no. 7, pp. 2442–2450, Jul. 2005.
[10] L. Miao, J. J. Zhang, C. Chakrabarti, and A. Papandreou-Suppappola, "A new parallel implementation for particle filters and its application to adaptive waveform design," in *Proc. IEEE Workshop Signal Process. Syst.*, 2010, pp. 19–24.
[11] O. Brun, V. Teulière, and J. Garcia, "Parallel particle filtering," *Elsevier J. Parallel Distrib. Comput.*, vol. 62, no. 7, pp. 1186–1202, 2002.
[12] V. Teulière and O. Brun, "Parallelisation of the particle filtering technique and application to doppler-bearing tracking of maneuvering sources," *Elsevier J. Parallel Comput.*, vol. 29, no. 8, pp. 1069–1090, 2003.
[13] A. S. Bashi, V. P. Jilkov, X. R. Li, and H. Chen, "Distributed implementations of particle filters," in *Proc. IEEE Int. Conf. Inf. Fusion*, 2003, pp. 1164–1171.
[14] B. Balasingam, M. Bolić, P. M. Djurić, and J. Miguez, "Efficient distributed resampling for particle filters," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2011, pp. 3772–3775.
[15] K. Achutegui and J. Míguez, "A parallel resampling scheme and its application to distributed particle filtering in wireless networks," in *Proc. IEEE Workshop Comput. Adv. Multi-Sensor Adaptive Process*., 2011, pp. 81–84.
[16] M. Chitchian, A. Simonetto, A. S. van Amesfoort, and T. Keviczky, "Distributed computation particle filters on GPU architectures for real-time control applications," *IEEE Trans. Control Syst. Technol.*, vol. 21, no. 6, pp. 2224–2238, Nov. 2013.
[17] G. Hendeby, R. Karlsson, and F. Gustafsson, "Particle filtering: The need for speed," *EURASIP J. Adv. Signal Proc.*, vol. 2010, 2010, Art. no. 181403.
[18] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*. London, U.K.: Chapman and Hall, 1994.
[19] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *J. Comput. Graphical Statist.*, pp. 1–25, 1996.
[20] J. Liu and R. Chen, "Sequential Monte Carlo methods for dynamic systems," *J. Amer. Stat. Assoc.*, vol. 93, pp. 1032–1044, 1998.
[21] J. Carpenter, P. Clifford, and P. Fearnhead, "An improved particle filter for nonlinear problems," *IEE Proc. Radar, Sonar Navigation*, vol. 146, pp. 2–7, 1999.
[22] P. B. Choppala, P. D. Teal, and M. R. Frean, "Soft systematic resampling for accurate posterior approximation and increased information retention in particle filtering," in *Proc. Workshop IEEE Statistic. Signal Process*., 2014, pp. 260–263.

[23] M. Bolić, P. M. Djurić, and S. Hong, "Resampling algorithms for particle filters: A computational complexity perspective," *EURASIP J. Appl. Signal Process.*, vol. 2004, pp. 2267–2277, 2004.
[24] L. Murray, "GPU acceleration of the particle filter: The metropolis resampler," 2012, *arXiv:1202.6163*.
[25] L. M. Murray, A. Lee, and P. E. Jacob, "Parallel resampling in the particle filter," *J. Comput. Graphical Statist.*, vol. 25, no. 3, pp. 789–805, 2016.
[26] O. Dulger, H. Oguztuzun, and M. Demirekler, "Memory coalescing implementation of metropolis resampling on graphics processing unit," *Springer J. Signal Process. Syst.*, vol. 90, no. 3, pp. 433–447, 2018.
[27] M. A. Nicely and B. E. Wells, "Improved parallel resampling methods for particle filtering," *J. IEEE Access*, vol. 7, pp. 47 593–47 604, 2019.
[28] A. Syed and O. Gustaffson, "Improved particle filter resampling architectures," *Springer J. Signal Process. Syst.*, pp. 1–14, 2019.
[29] X. Zhang and G. Feng, "Adaptive particle sampling and resampling architectures," in *Proc. IEEE Spring Simul. Conf.*, 2019, pp. 1–12.
[30] A. Varsi, J. Taylor, L. Kekempanos, E. Puzer, and S. Maskell, "A fast parallel particle filter for shared memory systems," in *Proc. IEEE Signal Proc. Lett.*, vol. 27, pp. 1570–1574, 2020.
[31] Q. K. Telesford, K. E. Joyce, S. Hayasaka, J. H. Burdette, and P. J. Laurienti, "The ubiquity of small-world networks," *J. Brain Connectivity*, vol. 1, no. 5, pp. 367–375, 2011.
[32] T. Li, M. Bolic, and P Djuric, "Resampling methods for particle filtering: Classification, implementation, and strategies," *IEEE Signal Proc. Mag.*, vol. 32, no. 3, pp. 70–86, May 2015.
[33] X. Hubaut, "Strongly regular graphs," *J. Elsevier*, vol. 13, no. 4, pp. 357–381, 1975.
[34] A. L. Barabási and R. Albert, "Emergence of scaling in random networks," *J. Amer. Assoc. Advance. Sci.*, vol. 286, no. 5439, pp. 509–512, 1999.
[35] R. A. Olshen and C. J. Stone, *Classification and Regression Trees*. Wadsworth Intl. Group, 1984.
[36] D. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," *J. Nat.*, vol. 393, no. 6684, pp. 440–442, 1998.
[37] P. Erdös and A. Rényi, "On the evolution of random graphs," *Publications Math. Inst. Hung. Acad. Sci.*, vol. 5, pp. 17–61, 1960.
[38] M. Krivelevich and B. Sudakov, "The phase transition in random graphs: A simple proof," *J. Random Struct. Algorithms*, 2012.
[39] B. Luque and R.V. Solé, "Phase transitions in random networks: Simple analytic determination of critical points," *Phys. Rev. E*, vol. 55, no. 1, pp. 257–260, 1997.
[40] N. Vaswani, "Particle filtering for large-dimensional state spaces with multimodal observation likelihoods," *IEEE Trans. Signal Process.*, vol. 56, no. 10, pp. 4583–4597, Oct. 2008.
[41] P. B. Choppala, P. D. Teal, and M. R. Frean, "Particle filter parallelisation using random network based resampling," in *Proc. IEEE Conf. Inf. Fusion*, 2014, pp. 1-8.
[42] R. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Eng.*, vol. 82, no. 1, pp. 35–45, 1960.
[43] F. J. Massey Jr, "The Kolmogorov-Smirnov test for goodness of fit," *J. Amer. Stat. Assoc.*, vol. 46, no. 253, pp. 68–78, 1951.
[44] X.-L. Hu, T. B. Schon, and L. Ljung, "A basic convergence result for particle filtering," *IEEE Trans. Signal Process.*, vol. 56, no. 4, pp. 1337–1348, Apr. 2008.
[45] M. Klaas, M. Briers, N. D. Freitas, A. Doucet, S. Maskell, and D. Lang, "Fast particle smoothing: If I had a million particles," in *Proc. ACM Int. Conf. Mach. Learn.*, 2006, pp. 481–488.

Praveen B. Choppala (Member, IEEE) received the master's degree in engineering with a Chancellor Medal from Karunya University, Coimbatore, India, in 2008, and the Ph.D. degree from the Victoria University of Wellington, New Zealand, in 2014. After his Ph.D., he was a Postdoc with the University of New South Wales, Sydney, NSW, Australia, where he developed fast Bayesian methods for time series analysis. He is currently an Associate Professor with the Department of Electronics and Communication Engineering, Wellfare Institute of Science Technology and Management, Visakhapatnam, India. His research interests include Bayesian estimation, Monte Carlo methods, and statistical signal processing. He was the recipient of the prestigious Industrial Research Limited Science Scholarship, New Zealand.

Marcus R. Frean received the B.Sc. degree from Massey University, Palmerston North, New Zealand, and the Ph.D. degree from the Theoretical Physics Group, Centre for Cognitive Science, Edinburgh University, Edinburgh, U.K. He has held a series of Postdoc positions with Cambridge University, Cambridge, U.K., Otago University, Dunedin, New Zealand, and Queensland University, Brisbane, QLD, Australia. Since 2000, he has been with the Victoria University of Wellington, Wellington, New Zealand, where he is currently a Senior Lecturer and an Associate Professor with the School of Engineering and Computer Science. In 2018, he was a Visiting Professor with the Max Planck Institute for Evolutionary Biology, Plon, Germany. His research interests include neural computation and Bayesian (probabilistic) inference.

Paul D. Teal (Senior Member, IEEE) received the B.E. degree with University Medal from the University of Sydney, Sydney, NSW, Australia, in 1990 and the Ph.D. degree from Australian National University, Canberra, ACT, USA, in 2002. He has been employed in Australia by Telstra, and in New Zealand by Concord Technologies, Caravel Consultants and Industrial Research Limited, in a number of technology development, deployment, consultant and research roles involving telecommunications infrastructure, industrial telemetry and control, voice processing, and call centres.

Since 2006, he has been a Senior Lecturer and an Associate Professor with the Victoria University of Wellington, Wellington, New Zealand. His research interests include theory and development of machine learning and Bayesian signal processing algorithms and their application to inverse problems in nuclear magnetic resonance, biomedical devices, audio, acoustics, perception of sound, generative models, such as normalizing flows, probabilistic circuits, and diffusion models.