Particle filter parallelisation using random network based resampling

Praveen B. Choppala, Paul D. Teal, Marcus R. Frean
School of Engineering and Computer Science
Victoria University of Wellington
New Zealand
Email: {praveen, pault, marcus}@ecs.vuw.ac.nz

***Abstract*—The particle filter approximation to the posterior density converges to the true posterior as the number of particles used increases. The greater the number of particles, the higher the computational load, which can be implemented by operating the particle filter in parallel architectures. However, the resampling stage in the particle filter requires synchronisation, extensive interchange and routing of particle information, and thus impedes the use of parallel hardware systems. This paper presents a novel resampling technique using a fixed random network. This idea relaxes the synchronisation constraints and minimises the particle interaction to a significant level. Using simulations we demonstrate the validity of our technique to track targets in linear and non-linear sensing scenarios.**

I. Introduction

The particle filter (PF) [1] is a Bayesian inference algorithm that provides a framework for target state estimation in non-linear and non-Gaussian scenarios. The main principle of the PF is to recursively generate a set of weighted particles which approximate the posterior pdf of the target state [2, 3]. The PF can be considered as a technique that uses two approaches in succession. The first, *sequential importance sampling (SIS)* specifies the process of predicting new particles at each time sample and updating their weights. SIS by itself, results in degeneracy — a problem in which ultimately only one particle has significant weight. This problem is overcome using a second stage, *resampling* [4, 5], that resamples and replaces the particles based on their weights so that the PF represents the posterior more accurately [6].

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 [7]. This complexity can be conveniently overcome by the use of graphics processing units (GPUs), field programmable gate arrays (FPGAs) and application specific integrated circuits (ASICs) that aid in massive data parallelism. While the SIS stage of the PF can be readily parallelised because the prediction and update steps do not require particle concurrency, the major impediment to the parallelisation of the PF is the resampling step. Resampling imposes the following constraints: a) *synchronisation* among all the particles leading to particle inter-dependence; b) extensive data exchange (*inter-particle interchange*) between the particles; c) the inter-particle interchange changes in a *volatile* way at each time step which prohibits the use of parallel hardware systems for the PF. The problem of adapting the resampling to parallel architectures is drawing increased attention recently from the research community.

Over the years, many processor architecture models have been proposed to adapt PFs to parallel systems. The idea is to either globally [8] or locally [9, 10] resample the particles that are distributed within a few processing elements (PEs). Methods that accelerate the PF operation within the PEs by exploiting some inherent GPU specific features have also been proposed, for example, see [11]. The performance of these techniques is highly dependent on the number of particles assigned to a PE. Moreover, the routing between the PEs is still volatile for all these architectures because the particles to be shared among the PEs are unknown.

At the algorithmic level, most of the research has related to deterministic resamplers [8]. The partial deterministic scheme [7, 9] resamples the particles when the weights are larger or smaller than suitably chosen thresholds. The performance of this system is sensitive to the values of these thresholds. Moreover, the procedure causes unnecessary loss of information contained in the small weights. This loss is reduced by soft resampling [12] that redistributes the discarded weight amongst the lower weight particles. However, these schemes still suffer from particle synchronisation problems during the normalisation and sorting processes. The stochastic resamplers operate by first normalising the weights, evaluating the cumulative sum of these weights and then finding a value of the sum greater than a random sample drawn from $\mathcal{U}(0, 1]$. The existing stochastic resamplers, namely the multinomial [13], the stratified [14], the residual [15], the systematic [2, 16] and the residual systematic [7], demand extensive and ever-changing inter-particle interaction and hence are unsuitable for parallel systems. In contrast, the Metropolis resampler [17] proposes that each particle requires the ratio of its weight and the weight of $B : B \ll I$ other particles to conduct resampling. Although this reduces particle synchronisation, the particle interchange is still volatile because there is no prior knowledge of the B particles with which each particle must be compared.

One other technique proposed to accelerate PF operation is the independent Metropolis Hastings (IMH) algorithm [18] which acts as a hybrid between PF resampling and the Markov Chain Monte Carlo (MCMC) sampling schemes. In this technique, to obtain the particle set corresponding to the current time index, we draw $I + I_b$ particles (where I is the total number of particles) at random with replacement from the particle set corresponding to the previous time step, pass them

through the proposal function to obtain the predicted states, and each particle is retained or replaced by its predecessor based on the acceptance ratio of their weights. The main advantage here is that the expensive resampling procedure is avoided. However, in practice, the filter requires a long burn-in time I_b for high accuracy; this in turn increases the computational cost linearly. Although the amount of particle interaction required now is only pairwise, the particles need to be synchronised during the prediction step and the entire replacement step. This leads to high computational delay. The technique has also been implemented using FPGAs [19].

Our main contribution in this paper is the proposal to use a *random network* [20] as a fixed resampling unit in the PF. This network, when employed in a PF, assists in effective communication between the particles with minimal data exchange. The main idea here is that we use a random network to provide each weighted particle with a fixed set of other particles with which to interact. Resampling is then performed locally amongst the weights of these particles to draw a single particle. The merit of this idea is that the resampling is performed among the particles that are minimally connected using a fixed topology. This will accelerate the PF operation and hence facilitates the use of a large number of particles which can then represent the posterior more accurately.

The rest of the paper is organised as follows. In section II, we set the notation and give a brief overview of the PF. In section III, we introduce random networks and describe how we propose to use the network as a resampling unit in the PF. We then give the evaluation results in section V and conclude in section VI.

II. Particle filtering

In this section, we fix the notation and present the mathematical formalism of the PF. The state of a target x_k at time k may be characterised by the parameters that define it, for example, its position, velocity, acceleration, etc. The target maneuvers under a Markovian motion model and the aim is to recursively obtain a best estimate of x_k using the noisy observations $z_{1:k}$ received from the sensors until the kth time step. Bayesian filtering [2] aids in this estimation by using the previous posterior $p(\mathrm{x}_{k-1}|z_{1:k-1})$ at time $k-1$ to evaluate the posterior $p(\mathrm{x}_k|z_{1:k})$ at time k. It is then easy to capture the information contained in x_k using $p(\mathrm{x}_k|z_{1:k})$. The Bayesian recursion is a two step procedure described by

- Prediction:

$$p(\mathrm{x}_k|z_{1:k-1}) = \int p(\mathrm{x}_k|\mathrm{x}_{k-1})p(\mathrm{x}_{k-1}|z_{1:k-1})d\mathrm{x}_{k-1} \tag{1}$$

- Update:

$$p(\mathrm{x}_k|z_{1:k}) = \frac{p(z_k|\mathrm{x}_k)p(\mathrm{x}_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \tag{2}$$

However, the integrals required to obtain (1) cannot usually be performed analytically and hence make the filtering process intractable.

The PF alleviates this problem by representing $p(\mathrm{x}_{k-1}|z_{1:k-1})$ by a set of weighted particles $\{\mathrm{x}^i_{k-1}, w^i_{k-1}\}^I_{i=1}$, where i is the particle index and I is the total number of particles. To obtain the representation of $p(\mathrm{x}_k|z_{1:k})$, the PF also operates in the two stages; a) prediction, and b) update. In the prediction stage, a new set of particles are sampled from an importance distribution as

$$\mathrm{x}^i_k \sim q(\mathrm{x}^i_k|\mathrm{x}^i_{k-1}, z_{1:k}) \tag{3}$$

and their weights are updated using

$$w^i_k = w^i_{k-1}\,\frac{p(z_k|\mathrm{x}^i_k)\;p(\mathrm{x}^i_k|\mathrm{x}^i_{k-1})}{q(\mathrm{x}^i_k|\mathrm{x}^i_{k-1}, z_{1:k})} \tag{4}$$

The posterior can be approximated using the *normalised* weights as

$$p(\mathrm{x}_k|z_{1:k}) \approx \sum_{i=1}^{I} w^i_k \delta(\mathrm{x}_k - \mathrm{x}^i_k) \tag{5}$$

After a few iterations, the discrepancy between the weights increases, causing degeneracy problems. One measure of efficiency in this context is the effective sample size, defined as

$$I_{\text{eff}} = \frac{1}{\sum_{i=1}^{I}(w^i_k)^2} \tag{6}$$

and low I_{eff} exhibits high degeneracy. The solution to degeneracy is to resample the particles with replacement whenever I_{eff} falls below a certain threshold. The goal is to eliminate particles that have negligible weights and replace them by copies of other particles that have larger weights, i.e., for $i = 1, ..., I$, we sample an index $j(i)$ distributed according to the probability $\Pr\{j(i) = m\} = \{w^m_k\}^I_{m=1}$ and replace $\mathrm{x}^i_k = \mathrm{x}^{j(i)}_k$ and set $w^i_k = 1/I$.

III. Random network based resampling

We now describe the process of generating random networks and then present our proposed resampling methodology using these random graphs.

A. Random networks

A graph is a representation of a set of objects among which a few (or all) are connected. These objects can be abstracted as *nodes* and their connections as *edges*. 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 connection between two nodes, we term the nodes *neighbors*. Let us consider a network having I nodes. We assume every node is connected to itself. The goal then is to form a network in which every node has $J-1$ neighbors such that the information on any node can be easily transmitted across all the nodes in minimum time. Random networks [20] are one of the networks that suit the above requirement. The main advantage of these graphs is that they can be constructed very simply. Let us denote the ith node as $\mathcal{V}_i$ and let $\rightarrow$ symbolise a directed edge between two nodes. Since we assume that every node is connected to itself, its probability function can be specified as

$$\Pr\{\mathcal{V}_i \rightarrow \mathcal{V}_i\} = 1 \tag{7}$$

The remaining $J-1$ neighbors can be chosen with uniform distribution

$$\Pr\{\mathcal{V}_i \rightarrow \{\mathcal{V}_j\}_{j\sim\mathcal{U}[1,I],j\neq i}\} = \frac{1}{I-1} \tag{8}$$

The maximum connectivity that the network offers with only a few neighbors is the property that defines the sparsity of the graph, i.e., most of the entries of its adjacency matrix $\mathbf{G}$ would be zero when $\mathbf{G}$ is defined by

$$\mathbf{G}_{i,j} = \begin{cases} 1 & \text{if } \mathcal{V}_i \rightarrow \mathcal{V}_j \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The parameter that regulates the sparsity of the structure is J. The larger the value of J, the faster the transmission of information among nodes. Small values of J (usually at $J \leq 2$) could result in islands of nodes such that the information contained in a node cannot reach every node although the probability of an island is very low, even with small J [21, 22].

B. Resampling using a random network

In this paper, we propose to treat a node as a particle and an edge between two nodes implies that electronic wiring connects the two particles. For I particles, we can form a random network such that $\mathcal{V}_i \rightarrow \mathcal{V}_j \Rightarrow \mathrm{x}_k^i \rightarrow \mathrm{x}_k^j$ and then construct the adjacency matrix $\mathbf{G}$ for the entire particle set. $\mathbf{G}$ which lists the set of neighbors (or connections) for each particle is computed only once. 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 particles are routed according to the topology described in $\mathbf{G}$, i.e., if the particle x_k^i has the set of $J-1$ particles $\{\mathrm{x}_k^j\}_{j=1,j\neq i}^{J-1}$ as its neighbors, then an electronic connection is made between x_k^i and its $J-1$ neighbors. These connections are fixed during the PF operation. We can then resample the particles in one of the two following procedures:

Deterministic resampling: A particle x_k^i and its $J-1$ neighbors are deterministically resampled by choosing the particle with the maximum weight and setting its weight to $1/I$. The key advantages of this scheme are that the data dependency constraints are extensively relaxed and the inter-particle interchange is limited to J particles instead of I. This accelerates the PF substantially. The only instance where synchronisation is required is when comparing the J particles to find the maximum weight. Moreover, the weights need not be normalised. However, since the technique is *greedy* and resamples only the best particles, the lower weight particles are mostly ignored and this could prove detrimental in high noise scenarios. This problem is overcome using stochastic resampling.

Stochastic resampling: A particle x_k^i and its $J-1$ neighbors can be resampled randomly 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 can then be reset to $1/I$. This procedure will ensure that small weights are not neglected all the time but at the expense of extra interaction between the particles during normalisation. The inter-particle routing, however, is now limited to only J particles that are still connected based on a fixed topology. This will in turn minimise the particle interchange and accelerate the PF operation.

IV. A COMPARATIVE DISCUSSION ON THE PROPOSAL

Here, we discuss the merits of the proposed random network based resampling over state-of-the-art resamplers in terms of particle interaction, volatility and synchronisation. This analysis is illustrated by constructing a matrix that provides a visualisation of the interactive relation between particles during the resampling step. For easy inspection of the matrix as an image, the number of particles is chosen to be $I = 256$. The black portion of the figures represent zeros and the white portion represent ones.

The inter-particle interaction for the proposed resampler can be described using the adjacency matrix $\mathbf{G}$ formed using (9). $\mathbf{G}$ for a random network with the number of interactions (or degree of connectivity) $J = 5$ is shown in Fig. 1. The ones along the diagonal indicate that every particle uses its own weight information to resample from its neighbors. Therefore, every particle needs to interact with only 4 *other* particles every time sample. This resampling structure is computed only once and remains *fixed* (non-volatile) throughout the PF procedure. Moreover, our technique does not require normalisation, and hence does not require synchronisation of all particles. We therefore claim *our proposition is highly suitable for implementation in parallel hardware architectures.*

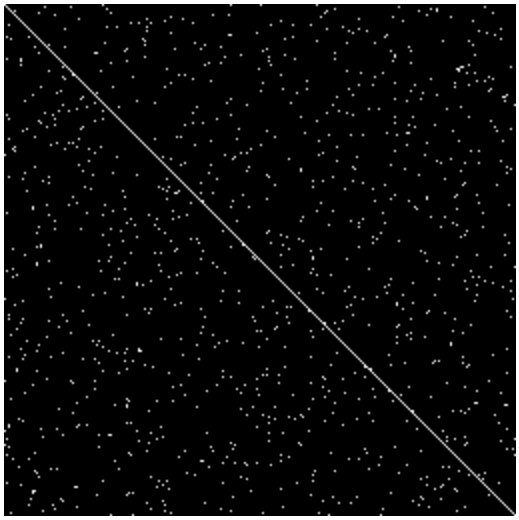

Fig. 1: The $\mathbf{G}$ matrix for random network based resampling. $\mathbf{G}$ is fixed for the entire PF operation. The number of particles $I = 256$. The number of interactions $J = 5$.

To discuss the conventional resampling schemes, we use Fig. 2, Fig. 3 and Fig. 4 that illustrate inter-particle interactive relation such that every particle along the vertical axis (256 particles totally) interacts with those particles along the horizontal axis whose index value is a one, i.e, the white portion.

The interactive relation for the Metropolis resampler [17] at a single time sample is shown in Fig. 2. Here we use $B = \log I = 8$. Since every particle interacts with only 8 other particles, the data exchange is minimal. Moreover, the technique does not require normalisation. However, the resampling structure changes every time sample. Consequently, the wiring between the particles is volatile and this causes

substantial delay in processing the incoming sensor data. The evaluation of the MCMC acceptance ratio BI times at each time step will induce additional computational delay. In light of these limitations, it is difficult to imagine an efficient hardware implementation for the Metropolis resampler.

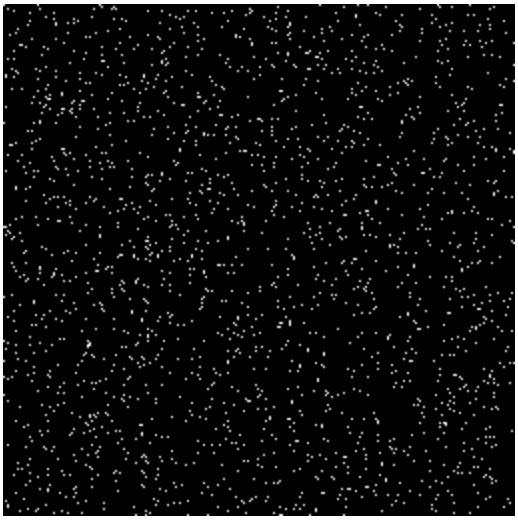

Fig. 2: The particle interactive relation in the Metropolis resampler at a single time sample. The number of particles $I = 256$. The number of interactions for each particle $B = \log I$.

The interactive relation for the multinomial resampler [13] and the systematic resampler [2, 16] at one time sample is shown in Fig. 3 and Fig. 4 respectively. The multinomial resampler [13] searches along the cumulative of the normalised weights from the starting value. As a result, the interaction is expensive (depicted by the white portion).

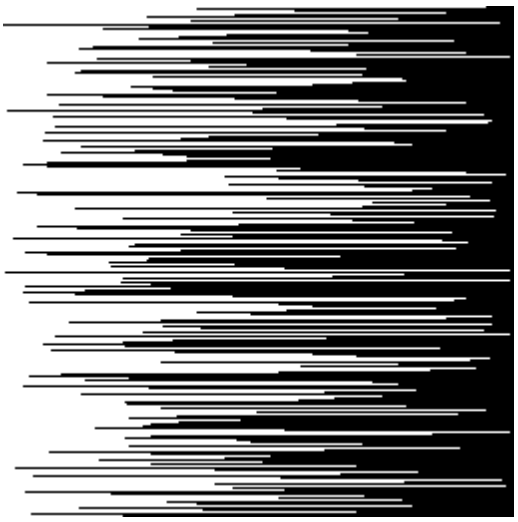

Fig. 3: The particle interactive relation in the multinomial resampler at a single time sample. The number of particles $I = 256$.

The systematic resampler [2] (and the stratified resampler [14]), in contrast, searches along the cumulative sum after pre-partitioning the space $(0, 1]$ into I linearly increasing disjoint sets. As a result, the data exchange between the particles is drastically reduced. However, both the systematic and the multinomial resampling techniques are volatile and hence the resampling structure will have to change every time sample. Besides, the techniques require normalisation and calculation of the weight cumulative sum.

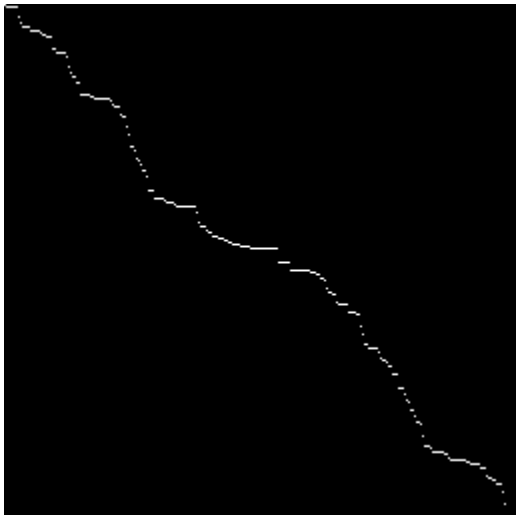

Fig. 4: The particle interactive relation in the systematic resampler at a single time sample. The number of particles $I = 256$.

The deterministic resamplers: the soft [12] and partial deterministic [7] resamplers, are comprised of operations like sorting and finding the maximum/minimum weights, etc., which require each particle to interact with all the other particles. Therefore, if one were to construct the inter-particle inter-dependency matrix for a deterministic resampler, it would contain all ones. Consequently, all the particles need to be synchronised. Moreover, these resamplers require the weights to be normalised.

A summary of the analysis and our claims is presented below.

	Interaction	Volatility	Normalisation	Sorting
Multinomial	High	Yes	Yes	No
Systematic	Low	Yes	Yes	No
Soft	High	Yes	Yes	Yes
Metropolis	Low	Yes	No	No
Proposed	**LOW**	**NO**	**NO**	**NO**

V. Evaluation

In this section, we first analyse the performance of both the variants of the proposed technique for a linear Gaussian model (for which the optimal posterior is known) in terms of their ability to accurately represent the posterior and to track with sufficient accuracy. We also show the performance of the proposed method at different levels of connectivity J. We

then examine the efficiency of the technique to track extended targets in the presence of false alarms and missed detections. Finally, the potential of the technique to accurately track targets in a non-linear sensing scenario is presented.

For ease of visualisation, we do not show all the possible existing resamplers. We found that the performance of the stratified, the residual and the residual systematic resamplers was almost identical to that of the systematic resampler and hence we only show the results for the latter. Moreover, the partial deterministic and the soft resamplers exhibit similar performance and hence we show only the last of these. This applies to all the figures presented in this paper.

For testing under linear Gaussian scenarios, we use a 1D model in which the state x_k is characterised by the target position. The process and observation models are

$$\mathrm{x}_k = F\mathrm{x}_{k-1} + q_{k-1} \tag{10}$$

$$z_k = H\mathrm{x}_k + r_k \tag{11}$$

where $F = 1$, and the uncertainty in the state evolution is $q_{k-1} \sim \mathcal{N}(0, 5)$, $H = 1$, and the sensor noise is $r_k \sim \mathcal{N}(0, \sigma^2)$ where $\sigma^2 = 0.5$. We assume a clean environment where an observation is available at every second, the observation is synchronised with time k and no clutter is observed. The results for this model are averaged over 20 experiments, each comprising 50 time steps[1].

Firstly, we test the faithfulness of our proposed techniques in representing the posterior in accordance to their Kolomogorov-Smirnov (KS) statistic [23] disagreement with the optimal Kalman filter [24]. KS testing has been used previously for other PF problems [25], and provides a reliable measure of the accuracy of the estimate of the posterior. KS testing for multi-dimensional Gaussian models can be conducted using the method in [26, 27]. In this paper, we use a different approach. After the resampling step, we de-mean and de-correlate the particles according to the theoretically optimal Kalman distribution and then take the maximum KS deviation.

Fig. 5 shows the KS statistic of the PFs operating on various resamplers for varying numbers of particles. It can be observed that while the *resource-hungry and parallel-unfriendly systematic resampler* exhibits the best performance, our *parallel-friendly and minimally interacting random 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 parallelisation is now possible. For example, the performance obtained by the systematic resampler at 32 and 64 particles can be achieved by using our proposed resampler at 64 and 256 particles respectively in the stochastic (non-greedy) variant, and 128 and 1024 particles respectively in the deterministic (greedy) variant. However, the performance of the deterministic variant becomes asymptotic for increasing J. The stochastic variant is more effective than that of the deterministic counterpart by virtue of leveraging the lower weight particles in the resampling process.

[1]A simulation video for this model is available at http://ecs.victoria.ac.nz/Groups/CSP/PublicationExtras

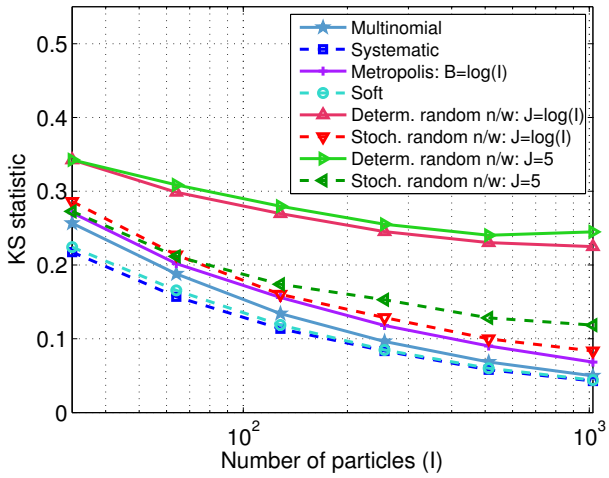

Fig. 5: KS statistic versus the number of particles. For the proposed resamplers, we use $J = \log(I)$ and 5. For the Metropolis resampler [17], we use $B = \log(I)$.

Fig. 6 shows the KS statistic of the deterministic and stochastic versions for varying degrees of connectivity J in the random network. The performance of the stochastic variant improves with increasing J. The performance of the deterministic variant improves with decreasing J and low values of J aid in better PF acceleration and easier parallelisation. At $J \leq 5$, the deterministic 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 deterministic 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.

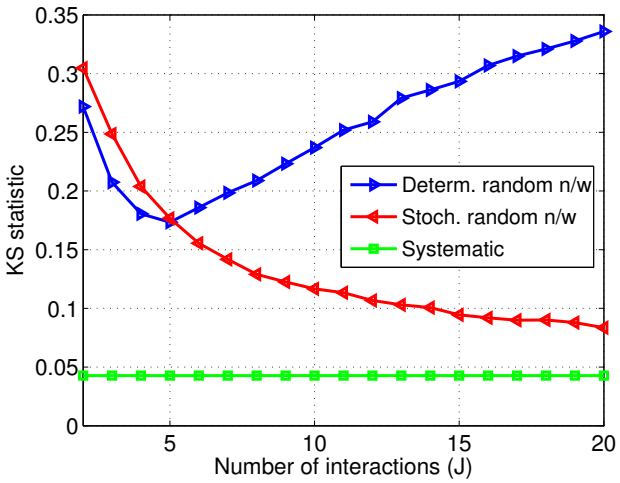

Fig. 6: KS statistic versus the number of interactions J. The number of particles $I = 1024$. J does not apply to the systematic resampler.

We now test the accuracy of the proposed technique for the same model in (10) and (11) by plotting the root mean

square error (rmse) for varying sensor noise variance σ^2. This is shown in Fig. 7. When comparing the proposed variants, the stochastic variant performs better by virtue of randomly resampling the particles, as opposed to the deterministic variant that *always* selects only the particle with the maximum weight. In terms of comparing the proposed technique with state-of-the-art resamplers, the stochastic variant matches the performance of the systematic resampler, while the deterministic variant tracks accurately at low noise levels.

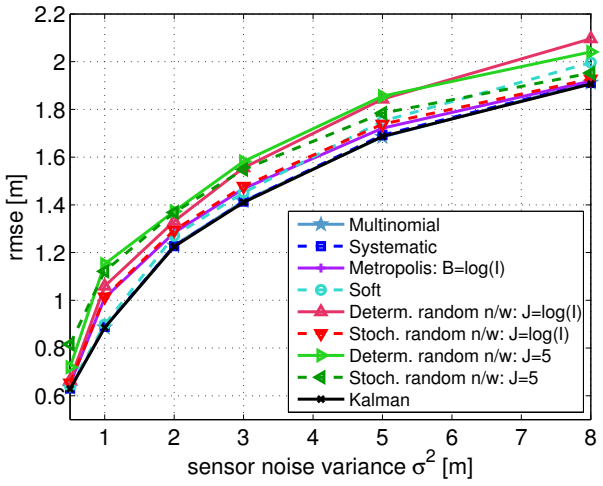

Fig. 7: rmse versus sensor noise variance σ^2. The number of particles $I = 1024$.

Fig. 8 shows the rmse of the proposed techniques for varying degree of connectivity J in the random network. At $J \lesssim 5$ and low σ^2, the deterministic variant exhibits better accuracy than that of the stochastic variant because its particle distribution is much closer to that of the optimal Kalman distribution as seen in Fig. 6. In high noise scenarios (large σ^2), we observe that the stochastic variant outperforms the deterministic variant because the latter *always* eliminates

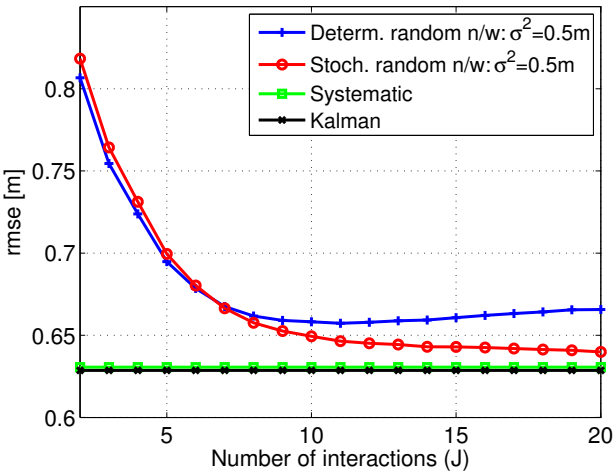

Fig. 8: rmse versus the number of interactions J. The number of particles $I = 1024$. J does not apply to the systematic resampler and the Kalman filter.

lower weight particles that may be slowly gaining weight, for example, when the target is making a sharp maneuver.

From the above analysis, it is observed that the proposed deterministic variant exhibits high fidelity in posterior representation and sufficient accuracy with fewer inter-partice interactions J and lesser sensor noise variance σ^2 respectively. However, at high J, the process suffers loss of particle diversity as only the large weight particles are retained. The performance of the stochastic resampler improves with increasing J because more particles are fed to the random resampling unit which then selects the lower weight particles with non-zero probability.

Secondly, we test the ability of the various filters to track an extended target in the presence of false alarms and missed observations for the same 1D linear Gaussian model. Since extended targets generate more than one measurement, the sensor model is now

$$Z_k = \{z_k^e\}_{e=1}^{E_k} \cup \{y_k^f\}_{f=1}^{F_k} \tag{12}$$

where the observations are normally distributed around the target center with variance $\sigma_e^2 = 0.5$m. The number of observations the target can generate is assumed to be discrete uniformly distributed as $E_k \sim \mathcal{U}[1, E_{\max}]$ and we choose $E_{\max} = 2$. Each z_k is obtained from (11). We consider that the false alarms $\{y_k^f\}_{f=1}^{F_k}$ at each time sample are uniformly distributed across the 1D surveillance region as

$$y_k^f \sim \mathcal{U}[x_{\min}, x_{\max}] \tag{13}$$

with $x_{\min} = 1$m and $x_{\max} = 100$m. The number of false alarms F_k is Poisson distributed with mean $\lambda = 5$. The observation set therefore now contains target detections and false alarms. We assume that the sensor will miss the target observations with a probability $1 - p_D$ where p_D is the detection probability. For this model, Fig. 9 shows the dependence of the rmse on $1 - p_D$. The results are averaged over 30 experiments, each comprising 80 time steps[2].

When the target observations are not available for some duration, the particles usually start to diverge. Once these observations start arriving and are consistent, the weights of the particles near the target start to increase. Unlike the deterministic resamplers, the stochastic resamplers do not have any memory of the goodness of a particle. Hence, they sense the change in the particle weights faster than the deterministic resamplers and thereby converge rapidly towards the true target location. It can be observed that our proposed resamplers and the Metropolis resampler aid in better accuracy. A possible explanation for this observation is that the divergence of the particles from the true target location (due to the information loss caused by the false alarms and the missed detections) is slower than that of the other techniques.

Finally, we test the accuracy of our techniques for tracking targets using a non-linear image sensing model. The model and the specifications used can be found in Ch.11 of [28]. The results for this model are averaged over 20 experiments, each comprising 30 time steps. We assume the target is present all the time[2]. Fig. 10 shows the track error in the PFs operating with state-of-the-art and the proposed resamplers for varying

[2]A simulation video for this model is available at http://ecs.victoria.ac.nz/Groups/CSP/PublicationExtras

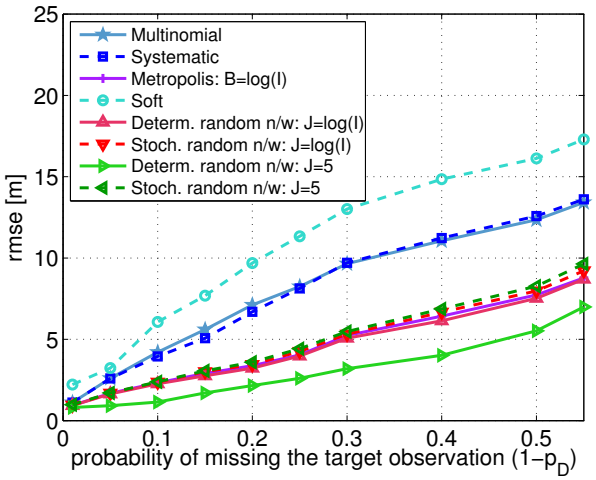

Fig. 9: rmse versus the probability that a sensor misses an observation. The number of particles $I = 1024$. The particles are initialised uniformly in the surveillance region, and a burn-in time of 10 seconds is then applied.

sensor noise levels. Among the conventional resamplers, the systematic exhibits high accuracy, but is outperformed by the proposed stochastic variant at high noise levels. A likely explanation for this is that choosing among *fewer* particles randomly based on their weights (the stochastic variant operation) makes the filter less responsive to spurious peaks in the image at high noise levels, as compared to choosing randomly among *all* the particles (the systematic and other resamplers). The track accuracy of the deterministic variant, although high for fewer inter-particle interactions J, deteriorates for increasing J and at high noise conditions. Futhermore, the stochastic variant exhibits better accuracy than that of the deterministic variant.

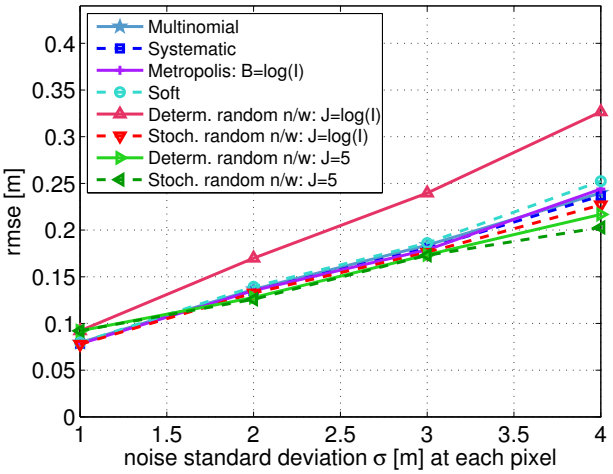

Fig. 10: rmse versus the noise standard deviation contributed by the staring camera at each pixel in the image. The number of particles $I = 1024$.

VI. Conclusion and Future Work

In this paper, we proposed the novel idea of using a random network as a fixed resampling unit for the PF. This network provides each particle with a fixed set of other particles with which it will interact and the resampler samples one particle from the set either deterministically or stochastically. Using simulations, we examined our methods regarding their ability to accurately represent the posterior. We then showed that our techniques can track accurately in linear and non-linear scenarios as well as cluttered environments. We also discussed the reduction in inter-particle interaction with the help of an particle interaction matrix. The key advantages of this proposal are that the particle dependency and the information interchange between the particles is significantly reduced. The principle benefit, however, is the ability to use a fixed topology that allows fixed electronic routing of the particles. This shows that the PF can now be accelerated by operating in parallel systems with more ease, and we can use more particles for a better posterior representation. In the future, we will aim to implement our idea in real time FPGA architectures and also investigate the potential of the techniques to perform under asynchronous sensor measurements.

References

[1] N. Gordon, D. Salmond, and A. Smith, "Novel approach to non-linear/non-Gaussian Bayesian state estimation," in *Proc. IEE Radar and Signal Process.*, vol. 140, no. 2, 1993, 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, 2002.

[3] P. Djurić, J. Kotecha, J. Zhang, Y. Huang, T. Ghirmai, M. F. M. Bugallo, and J. Miguez, "Particle filtering," *IEEE Signal Process. Mag.*, vol. 20, no. 5, pp. 19–38, 2003.

[4] J. Hol, T. Schön, and F. Gustafsson, "On Resampling Algorithms for Particle Filters," in *Proc. Nonlinear Stat. Signal Process. Workshop*, 2006.

[5] R. Douc and O. Cappé, "Comparison of resampling schemes for particle filtering," in *Proc. Int. Symp. Image and Signal Process. and Analysis*, 2005, pp. 64–69.

[6] A. Doucet, N. De Freitas, and N. Gordon, *Sequential Monte Carlo methods in practice*. Springer New York, 2001, vol. 1.

[7] M. Bolić, P. Djurić, and S. Hong, "Resampling algorithms for particle filters: A computational complexity perspective," *EURASIP J. of Applied Signal Process.*, vol. 2004, pp. 2267–2277, 2004.

[8] O. Brun, V. Teulière, and J. Garcia, "Parallel particle filtering," *Elsevier J. Parallel and Distributed Computing*, vol. 62, no. 7, pp. 1186–1202, 2002.

[9] M. Bolić, P. Djurić, and S. Hong, "Resampling algorithms and architectures for distributed particle filters," *IEEE Trans. Signal Process.*, vol. 53, no. 7, pp. 2442–2450, 2005.

[10] A. Bashi, V. Jilkov, X. Li, and H. Chen, "Distributed implementations of particle filters," in *Proc. IEEE Int. Conf. Information Fusion*, 2003, pp. 1164–1171.

[11] G. Hendeby, R. Karlsson, and F. Gustafsson, "Particle filtering: the need for speed," *EURASIP J. Adv. Signal Proc.*, vol. 2010, p. 22, 2010.

[12] P. Choppala, P. Teal, and M. Frean, "Soft resampling for improved information retention in particle filtering," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, vol. 13, 2013, pp. 4036–4040.

[13] B. Efron and R. Tibshirani, "An introduction to the bootstrap," *Chapman and Hall*, 1994.

[14] G. Kitagawa, "Monte Carlo filter and smoother for non-Gaussian nonlinear state space models," *J. of Computational and Graphical Statistics*, pp. 1–25, 1996.

[15] J. Liu and R. Chen, "Sequential Monte Carlo Methods for Dynamic Systems," *J. American Statistical Asso.*, vol. 93, pp. 1032–1044, 1998.

[16] J. Carpenter, P. Clifford, and P. Fearnhead, "An improved particle filter for nonlinear problems," *IEE Proc. Radar, Sonar and Navigation*, vol. 146, pp. 2–7, 1999.

[17] L. Murray, "GPU Acceleration of the Particle Filter: The Metropolis resampler," *arXiv preprint arXiv:1202.6163*, 2012.

[18] A. Sankaranarayanan, R. Chellappa, and A. Srivastava, "Algorithmic and Architectural design methodology for particle filters in hardware," in *IEEE Proc. Int. Conf. Computer Design*, 2005, pp. 275–280.

[19] 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 *IEEE Workshop Signal Process. Systems*, 2010, pp. 19–24.

[20] P. Erdős and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hungar. Acad. Sci.*, vol. 5, pp. 17–61, 1960.

[21] M. Krivelevich and B. Sudakov, "The phase transition in random graphs: A simple proof," *J. Random Structures & Algorithms*, 2012.

[22] B.Luque and R.V.Solé, "Phase transitions in random networks: simple analytic determination of critical points," *Physical Review E*, vol. 55, no. 1, pp. 257–260, 1997.

[23] J. A. Peacock, "Two-dimensional goodness-of-fit testing in astronomy," *Royal Astronomical Society*, vol. 202, pp. 615–627, 1983.

[24] R. Kalman, "A new approach to linear filtering and prediction problems," *J. Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[25] P. Djurić and J. Míguez, "Assessment of Nonlinear Dynamic Models by Kolmogorov–Smirnov Statistics," *IEEE Trans. Signal Process.*, vol. 58, no. 10, pp. 5069–5079, 2010.

[26] A. Justel, D. Peña, and R. Zamar, "A multivariate Kolmogorov-Smirnov test of goodness of fit," *Elsevier Statistics & Probability Letters*, vol. 35, no. 3, pp. 251–259, 1997.

[27] G. Fasano and A. Franceschini, "A multidimensional version of the Kolmogorov-Smirnov test," *Royal Astronomical Society*, vol. 225, pp. 155–170, 1987.

[28] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House Publishers, 2004.