Signal Processing 119 (2016) 115–127

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Algorithm design for parallel implementation of the SMC-PHD filter

Tiancheng Li [a,b,*], Shudong Sun [b], Miodrag Bolić [c], Juan M. Corchado [a,d]

[a] The BISITE group, Faculty of Science, University of Salamanca, 37008 Salamanca, Spain
[b] School of Mechanical Engineering, Northwestern Polytechnical University, 710072 Xi'an, PR China
[c] School of Electrical Engineering and Computer Science, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5
[d] Osaka Institute of Technology, Asahi-ku Ohmiya, 535-8585 Osaka, Japan

ARTICLE INFO

Article history:
Received 28 November 2014
Received in revised form
7 July 2015
Accepted 16 July 2015
Available online 4 August 2015

Keywords:
Multi-target tracking
Particle filter
Probability hypothesis density filter
Parallelization

ABSTRACT

The sequential Monte Carlo (SMC) implementation of the probability hypothesis density (PHD) filter suffers from low computational efficiency since a large number of particles are often required, especially when there are a large number of targets and dense clutter. In order to speed up the computation, an algorithmic framework for parallel SMC-PHD filtering based on multiple processors is proposed. The algorithm makes full parallelization of all four steps of the SMC-PHD filter and the computational load is approximately equal among parallel processors, rendering a high parallelization benefit when there are multiple targets and dense clutter. The parallelization is theoretically unbiased as it provides the same result as the serial implementation, without introducing any approximation. Experiments on multi-core computers have demonstrated that our parallel implementation has gained considerable speedup compared to the serial implementation of the same algorithm.

1. Introduction

Multi-target tracking (MTT) involves the joint estimation of the number of multiple targets and their states in the presence of spontaneous birth/spawn/death of targets and clutter. MTT has a long history of research over a half of century, with many applications in both military and commercial realms [1]. Apart from handling the noises in the state dynamics and observation models, one has to contend with many more challenges, such as: (i) The number of targets is unknown and time varying because of the spontaneous birth, spawn and death of targets; (ii) clutter exists and can be significant; (iii) targets can be miss-detected; (iv) most challenging, data association between observations and targets in the presence of clutter that is required in traditional trackers is difficult.

The states of targets and observations in such an environment are finite-set-valued random variables that are random in both the number of elements and the values of the elements. The idea of modelling the states and observations as random finite set (RFS) is natural and it allows for overcoming the difficulty of data association [2,3] in the filtering stage. With the incorporation of RFS and point process theory in the MTT problem, the probability hypothesis density (PHD) filter provides a concise and tractable alternative to the optimal Bayesian filter that works by propagating the first-order moment associated with the multi-target Bayesian posterior and is essentially a density estimator.

* Correspondence to: Edificio I+D+I, C/ Espejo, s/n, 37008 Salamanca, Spain (Office). Tel.: +34 923 012 045; mobile: +34 655 24 8188; fax: +34 923 294 514.
E-mail addresses: tiancheng.li1985@gmail.com, t.c.li@usal.es (T. Li), sdsun@nwpu.edu.cn (S. Sun), mbolic@site.uottawa.ca (M. Bolić), corchado@usal.es (J.M. Corchado).

http://dx.doi.org/10.1016/j.sigpro.2015.07.013
0165-1684/

The PHD filter is attracting increasing attention, motivating different derivations, interpretations and implementations. It is found that the PHD filter asymptotically behaves as a mixture of Gaussian components, whose number is the true number of targets, and whose peaks collapse in the neighbourhood of the classical maximum likelihood estimates, with a spread ruled by the Fisher information [4]. A connection between the PHD recursion and spatial branching processes is established in [5], which gives a generalized Feynman–Kac systems interpretation of the PHD recursive equations and enables the derivation of mean-field implementations. A physical interpretation of the PHD filter based on the bin model is given in [6] which gives a more intuitive understanding of the PHD filter. The PHD filter has also been extended to solve more MTT-centric complex problems such as distributed sensor localization [7], parameter estimation [8] and mobile robot simultaneous localization and mapping [9], to name a few. Presently, the PHD filter has been implemented in forms of weighted particles [10], finite Gaussian mixtures (GM) [11] or their hybrids [12]. In particular, the sequential Monte Carlo (SMC) implementation that is often referred to the SMC-PHD filter is gaining special attention [13–15], which has also prompted new developments of SMC based on the RFS framework e.g. [16].

The advantage of SMC over GM is that it is free of linear and Gaussian assumptions. However, to maintain a sufficient approximation accuracy, a large number of particles are usually required causing a heavy computational burden. The situation gets worse in the SMC-PHD filter where the computational requirements also grow with the number of targets/observations [17]. Thence, fast computing techniques such as gating [18,19] and parallel processing [20–23] appear as promising approaches to ensure real-time performance, which however are often based on significant approximations. With the fast development of multi-core and multi-threading compatible hardware and software, the parallelization becomes increasingly attractive and even necessary. To the best of our knowledge, a fully parallel implementation of the SMC-PHD filter that is able to provide the same result as the serial implementation is still lacking. Such a parallel SMC-PHD filter is the focus of this paper.

There are four main steps in the implementation of particle filters (PFs) including state updating, weight updating, resampling and estimate extraction. The essence of parallelization is distributing calculation operations to different processing elements (PEs) for parallel computing. In the conventional parallel implementation of particle filters, particles are distributed among PEs. However, even for the PF targeted for single target tracking (referred to the basic PF hereafter), the resampling prevents direct parallel processing due to joint processing of all particles. The parallelization technique for resampling achieved in these particle filters e.g. [24–25] may be applied in the MTT-based particle PHD filter as they have been done in [20–22] at the price of inevitable considerable communication overhead. However, there are more challenging operations that consist of joint processing of particles in the SMC-PHD filter in addition to the resampling. First, the weight updating of each particle in the PHD updater requires the weight information of all the other particles thereby preventing the parallelization that distributes particles among PEs. Secondly, what used in [20–23] for extracting multiple estimates in the SMC-PHD filter is still the (k-means) clustering method which is computationally intensive and is less suitable for parallelization. It is unclear how the k-means clustering can be parallelized without introducing significant communication overhead. These operations together with the resampling step form the primary challenges for parallelization of the SMC-PHD filter.

To overcome the parallel computing difficulty in resampling, weight updating and estimate extraction, these steps are parallelized in a novel manner that is based on distributing observations instead of distributing particles while the prediction is carried out on the distributed resampling particles in each PE. In particular, new resampling and estimate extraction methods that are suitable for parallel processing are proposed. Significantly different from [20–23], our parallelization is able to obtain the same estimation result as the serial implementation. All four steps of the SMC-PHD filter are fully paralleled and significant speedup is achieved. The parallel algorithm is described with regard to the multi-core computer that consists of one central unit (CU) and several PEs.

The paper is arranged as follows. A novel insight of the SMC-PHD filter that partitions the weight of particles into components with respect to observations is provided in Section 2, which forms the foundation of our approach. Related works are also briefly described in this section. The technical details of our approach are presented in Section 3. Qualitative analysis and quantitative experiments are given in Sections 4 and 5 respectively. We conclude in Section 6.

2. Background and related work

2.1. An observation-based view of the PHD equation

To model the discrete filtering problem, the state is assumed to follow a Markov process in the state space $\chi \subseteq \mathbb{R}^{n_x}$, with a transition density $f_{k|k-1}(\cdot|\cdot)$. That is, a given state x_{k-1} at time $k-1$ will either die with probability $1-p_{S,k}(x_{k-1})$ or continue to exist at time k with survival probability $p_{S,k}(x_{k-1})$ and move to a new state with a transition probability density $f_{k|k-1}(x_k|x_{k-1})$. The Markov process is partially observed in the observation space $\mathcal{Z} \subseteq \mathbb{R}^{n_z}$. That is, at time k, a given target $x_k \in X_k$ is either miss-detected with probability $1-p_{D,k}(x_k)$ or detected with detection probability $p_{D,k}(x_k)$ and generates an observation $z_k \in Z_k$ with likelihood $g_k(z_k|x_k)$.

The collections of target states and observations at scan k can be represented as finite sets $X_k = \{x_{k,1}, \ldots, x_{k,N_k}\} \in F(\chi)$ and $Z_k = \{z_{k,1}, \ldots, z_{k,M_k}\} \in F(\mathcal{Z})$, where N_k and M_k are the number of targets and the number of observations respectively, while $F(\chi)$ and $F(\mathcal{Z})$ are the collections of all finite subsets of targets and observations, respectively.

Based on the finite set statistics, the PHD filter derived by Mahler estimates jointly the number and the state of targets by propagating in time the intensity function (namely the PHD function) [2]. The following assumptions are required in the PHD filter: (A.1) each target is assumed to evolve and generate observations independently of others; (A.2) the clutter distribution is assumed to be

Poisson and independent of the observations; (A.3) one target can generate no more than one observation at each scan; and (A.4) the appearing target process is Poisson. Let $D_{k|k}$ and $D_{k|k-1}$ be the intensity (PHD) functions associated to the posterior point processes $f_{k|k}(x_k|Z_{1:k})$ a nd $f_{k|k-1}(x_k|Z_{1:k-1})$, namely $D_{k|k}=D_{k|k}(x_k|Z_{1:k})$, $D_{k|k-1}=D_{k|k-1}(x_k|Z_{1:k-1})$. The PHD filter propagates the following Bayesian recursions connected by predictors and updaters:

$$\ldots\rightarrow D_{k-1|k-1}\rightarrow D_{k|k-1}\rightarrow D_{k|k}\rightarrow\ldots$$

The PHD predictor (state updating) is

$$D_{k|k-1}=\int_{\chi}\phi_{k|k-1}(x|u)D_{k-1|k-1}(u)du+\gamma_k(x) \tag{1}$$

where the following abbreviation is used:

$$\phi_{k|k-1}(x|u)=p_{S,k}(u)f_{k|k-1}(x|u)+b_k(x|u)$$

where $b_k(x|u)$ denotes the intensity function of the RFS $B_k(x|u)$ of targets spawned from the previous state u, and $\gamma_k(x)$ is the birth intensity function of new targets at scan k.

The PHD updater (weight updating) is

$$D_{k|k}(x)=\left(1-p_{D,k}(x)+\sum_{z\in Z_k}\frac{p_{D,k}(x)g_k(z|x)}{\kappa_k(z)+C_k(z)}\right)D_{k|k-1}(x) \tag{2}$$

where $\kappa_k(z)$ denotes the clutter intensity at time k and

$$C_k(z)=\int p_{D,k}(u)g_k(z|x)D_{k|k-1}(u)du \tag{3}$$

The SMC-PHD filter uses a set of particles to approximate the PHD predictor and updater. Given the importance densities $p_k(\cdot|Z_k)$ (for new born particles), $q_k(\cdot|x_{k-1},Z_k)$ (for existing particles) and assuming that there are L_{k-1} particles in time step $k-1$ (which does not apply for the initial step of the filter) and that J_k new particles are allocated for possible new-born targets, the PHD predictor $D_{k|k-1}$ can be written as

$$D_{k|k-1}(x_k)=\sum_{i=1}^{L_{k-1}+J_k}w_{k|k-1}^{(i)}\delta_{x_k^{(i)}}(x_k) \tag{4}$$

where the particle state and weight are given as

$$x_k^{(i)}\sim q_k\left(\cdot\left|x_{k-1}^{(i)},Z_k\right.\right),\quad i=1,\ldots,L_{k-1} \tag{5a}$$

$$x_k^{(i)}\sim p_k(\cdot|Z_k),\quad i=L_{k-1}+1,\ldots,L_{k-1}+J_k \tag{5b}$$

$$w_{k|k-1}^{(i)}=\frac{\phi_{k|k-1}\left(x_k^{(i)}\left|x_{k-1}^{(i)}\right.\right)w_{k-1}^{(i)}}{q_k\left(x_k^{(i)}\left|x_{k-1}^{(i)},Z_k\right.\right)},\quad i=1,\ldots,L_{k-1} \tag{6a}$$

$$w_{k|k-1}^{(i)}=\frac{\gamma_k\left(x_k^{(i)}\right)}{J_k p_k\left(x_k^{(i)}|Z_k\right)},\quad i=L_{k-1}+1,\ldots,L_{k-1}+J_k \tag{6b}$$

where $\delta_x(\cdot)$ denotes the delta-Dirac mass located in x.

The SMC-PHD updater $D_{k|k}$ can be written as

$$D_{k|k}(x_k)=\sum_{i=1}^{L_{k-1}+J_k}w_{k|k}^{(i)}\delta_{x_k^{(i)}}(x_k) \tag{7}$$

where

$$w_{k|k}^{(i)}=\left(1-p_{D,k}\left(x_k^{(i)}\right)+\sum\nolimits_{z\in Z_k}\frac{p_{D,k}\left(x_k^{(i)}\right)g_k\left(z\left|x_k^{(i)}\right.\right)}{\kappa_k(z)+C_k(z)}\right)w_{k|k-1}^{(i)} \tag{8}$$

$$C_k(z)=\sum_{j=1}^{L_{k-1}+J_k}p_{D,k}\left(x_k^{(j)}\right)g_k\left(z|x_k^{(j)}\right)w_{k|k-1}^{(j)} \tag{9}$$

Correspondingly, we define its decomposition unit as

$$c_k(z,j):=p_{D,k}\left(x_k^{(j)}\right)g_k\left(z|x_k^{(j)}\right)w_{k|k-1}^{(j)} \tag{10}$$

Also, the weight of particles can be divided as

$$w_k(z,j):=\begin{cases}\frac{c_k(z,j)}{\kappa_k(z)+C_k(z)}, & z\in Z_k\\ \left(1-p_{D,k}\left(x_k^{(j)}\right)\right)w_{k|k-1}^{(j)}, & z=z_0\end{cases} \tag{11}$$

where the symbol z_0 is introduced for consistency to represent the missed observation(s). $w_k(z,j)$, $z\in\{z_0,Z_k\}$ are referred to as weight components in the paper. Straightforwardly, we have

$$w_{k|k}^{(j)}=\sum_{z\in\{z_0,Z_k\}}w_k(z,j) \tag{12}$$

The sum of the weight components for each $z\in\{z_0,Z_k\}$ can be defined as

$$W_k(z):=\sum_{j=1}^{L_{k-1}+J_k}w_k(z,j) \tag{13}$$

Obviously, we have

$$\sum_{j=1}^{L_{k-1}+J_k}w_{k|k}^{(j)}=\sum_{z\in\{z_0,Z_k\}}W_k(z) \tag{14}$$

The PHD represents the local density of the distribution of targets, and its integration in a region corresponds to the expected number of targets in that region. In the SMC implementation, the expected number $\hat{N}_k$ of targets can be determined based on the total weight mass $W_k=\sum_z W_k(z)$ as

$$\hat{N}_k=[W_k]=\left[\sum\nolimits_{z\in\{z_0,Z_k\}}W_k(z)\right] \tag{15}$$

where the operator $[\bullet]$ rounds the content to the nearest integer.

Theorem 1. $\forall z\in Z_k$: $0\le W_k(z)\le 1$

Proof. By substituting (11) into (13) under the condition $z\in Z_k$, $W_k(z)$ can be expanded as

$$W_k(z)=\sum_{i=1}^{L_{k-1}+J_k}\frac{c_k(z,i)}{\kappa_k(z)+\sum_{j=1}^{L_{k-1}+J_k}c_k(z,j)}\in[0,1]$$

This only occurs when $p_D(\cdot)=0, W_k(z)=0$ and only when $\kappa_k(z)=0, W_k(z)=1$.

The relationship among components $w_k(z,j)$, $W_k(z)$ and $w_{k|k}$ can be depicted as shown in Fig. 1, which provides an insight of the weight of particles that is contributed jointly by the previous weight (i.e. the part $w_k(z_0,j)$) and by the

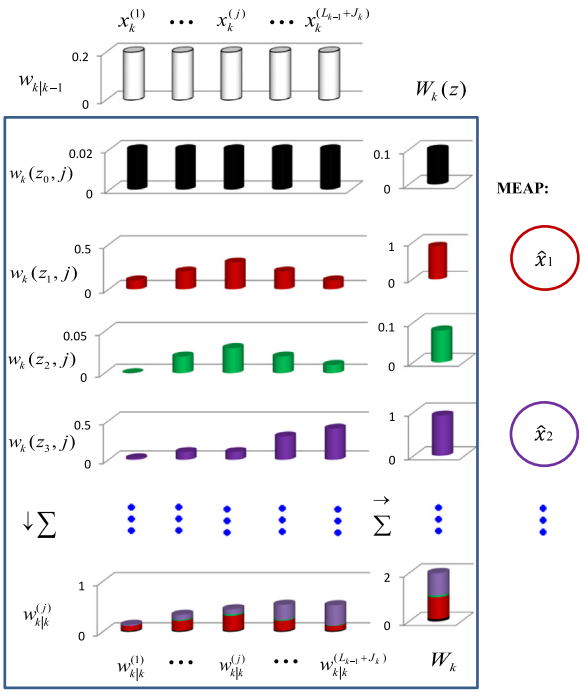

Fig. 1. Visualization of the decomposition of the weight of particles.

underlying observations (the part $w_k(z,j)$, $z \in Z_k$). The height of the cylinder represents the weight value, and the colour indicates different components $z \in \{z_0, Z_k\}$. As an example, the figure shows that large weight components related to z_1 and z_3 correspond to two potential targets (see the following subsection). It is necessary to note, our parallelization algorithm uses weight components $w_k(z,j)$ and $W_k(z)$, $z \in \{z_0, Z_k\}$ for updating and resampling of particles, and uses $W_k(z), c_k(z,j)$ and $C_k(z)$, $z \in Z_k$ for estimate extraction, with regard to the distributed observations.

2.2. Estimate extraction

The SMC-PHD filter, however, has not provided any solution for multiple estimate extraction; instead, a separate multiple estimate extraction method is required. There are two main types of solutions that differ based on whether the estimate extraction relies on the whole weight of the particles or on their components. In the first class that is based on the whole weight of particles, one of the most common means is clustering. However, clustering is essentially an iteration process that is computationally intensive and unreliable. Furthermore, the clustering does not scale well with the number of targets, and the performance of clustering depends on the estimation quality of the number of targets. The unreliability of clustering techniques for estimate extraction eschews the SMC-PHD filter, not to mention its slow computation. More importantly, most clustering algorithms are inherently serial and are hard to be parallelized.

The other group of estimate extraction methods is free of clustering and is executed with regard to individual observations including the methods proposed in [26–28] and the multi-EAP (Expected a Posterior, MEAP) method [29,30]. The original version of the MEAP estimator is given in Algorithm 1, in which $\Xi(a)$ is the RFS of particles that are associated to observation a based on the near and nearest neighbor (NNN) principle (i.e. each particle is associated to its nearest observation and also the observation lying in a validation area specified by a gate around the particle). For simplicity, $\Xi(a)$ can be defined as the whole particle set which will not affect the result much in most general cases (but can significantly save computation), namely abandoning the NNN association, otherwise one needs to associate particles to observations in advance before parallel computing. This allows direct parallelization based on distributed observations in our approach. The MEAP (without the NNN association), which is shown to be more accurate, computationally fast and reliable than clustering, will be incorporated into our parallelization framework with a slight change.

Algorithm 1. Multi-EAP (MEAP)

FOR $j = 1, \ldots, \min(\hat{N}_k, |Z_k|)$ DO

$$a = \arg\max_z \{W_k(z)\}_{z=1}^{|Z_k|} \tag{16}$$

$$x_j^{EAP} = \frac{\sum_{i \in \Xi(a)} g_k\left(a|x_k^{(i)}\right) w_{k|k-1}^{(i)} x_{k|k-1}^{(i)}}{\sum_{i \in \Xi(a)} g_k\left(a|x_k^{(i)}\right) w_{k|k-1}^{(i)}} \tag{17}$$

$W_k(a) = 0$

2.3. Parallelization challenges

As shown in the calculation for the PHD updater, there are many joint processing operations of all particles which prevent the traditional parallelization. As such, quite less work has been reported on parallel processing of the SMC-PHD filter; instead, there has been related work where parallel processing is attempted even though parallelization was not the main focus. For example, a multiple-model SMC-PHD filter capable of handling uncertain target dynamics is proposed in [31]. In this case, several filters, each matched to a different target motion mode, operate in parallel, and the overall state estimate is given by a weighted sum of the estimates from each filter.

More directly, the distributed resampling algorithms e.g. [24,25,32] that are based on dividing particles among PEs can be applied in the SMC-PHD filter. This enables some level of parallelization in the form of dividing particles, as with the basic PF [20]. Further understanding that the resampling does not need normalization in the SMC-PHD filter enables a higher level parallelization of the SMC-PHD filter [21]. To overcome the hardware challenge in field programmable gate array (FPGA) platforms that require a fixed number of observation processing elements, only selected observations were used while the others were abandoned [22]. A similar operation has been carried out in the simplified updating of particles [20] while possible independences are assumed between groups of particles that are associated to targets that are distant from each other in the state space. It is critical to note that these ad-hoc approaches may result in a change of the PHD equations and often suffer from approximation. As a consequence, approaches presented in [20–23] are

not able to obtain the same result as the serial implementation. In addition to resampling, there are more challenging operations to handle for parallelization: (i) Updating the weight of each particle requires the likelihood of other particles. (ii) Multiple estimate-extraction by using methods such as clustering is much more time-consuming than the single estimate-extraction in the basic PF. Together with the resampling step, these are the primary barriers for the parallelization of the SMC-PHD filter. Since the two critical steps of resampling (using known threshold-based or approximated sampling method) and the estimate extraction (using the serial k-means clustering) have not been parallelized properly in [20–22], the speedup obtained is primarily from gating rather than parallel processing. More importantly, so far there is no parallel approach that is able to obtain the same result as serial implementation.

Outside of the focus of this paper, distributed multi-target tracking based on a distributed sensor network is also becoming an important problem, where each node runs its own PHD estimator (generally one filter per node) and generally "consensus" is pursued among the nodes. For this special concern, readers are referred to [7,33–35] and [36,37]. Particularly, the parallel GM implementation of the single cluster (SC) PHD filter is proposed based on the graphics processing unit (GPU) [38]. Different to the general PHD filter, the Gaussian mixture SC-PHD filter propagates the parent state as a collection of particles, with each particle being linked to a daughter GM PHD conditioned on the state and trajectory of that particle. Our approach is also different to [39] which decomposes the state of particles into two parts.

Our approach focuses on the algorithm design and is not hardware-specific. To achieve high-level of parallelism, we develope parallel algorithms separately for all the four steps of the SMC-PHD filter. The prediction step is parallelized in terms of distributing particles achieved by the resampling; the weight updating, resampling and estimate extraction steps are parallelized in terms of distributing observations. This paradigm of two different distributing solutions is novel and critical for obtaining unbiased estimates and achieving full parallelization. In addition, new resampling and estimate extraction methods that are suitable for parallel processing are proposed. All of these make the significant and distinguishing difference of our parallelization framework to existing works.

3. Parallelization of the SMC-PHD filter

This section will detail the parallel algorithm design with respect to the four steps of the SMC-PHD filter.

3.1. Parallelizing computations

Our Parallel model is based on the typical centralized distributed system that consists of one CU and several independent PEs in which there is bidirectional communication between PEs and the CU and no communication among PEs. Therefore, our system represents the master-slave programming model and as such it is convenient for implementation on contemporary multi-core computers. Henceforth, computational units that need to be processed in parallel are called tasks. Parallel tasks are primarily carried out by distributed PEs and the serial computations by the CU that also manages task allocations and communication with PEs. One of the goals of our approach is to minimize the serial computation and communication between PEs and the CU in order to maximize parallel computing.

In this paper, the number of active processors may change with the number of observations. Let us assume that there are M tasks that need to be allocated to no more than P PEs. In our case, the parallel filter aims to use as many PEs as possible for maximum speed up. However, there is possibility that the number of tasks is less than the PEs, i.e. $M < P$, which will make some PEs idle. There will be either an equal number of tasks per PE or the number of tasks per PE can differ by one if it is not divisible by P.

The algorithmic representation of the proposed parallel framework for the SMC-PHD filter is given in Fig. 2. The left column indicates the steps processed by PEs, while the information in the blue rectangle on the right inside indicates the steps processed by the CU. All of the operations are processed in order from top to bottom, whereas operations at the same time level are processed in parallel. Communication between PEs and the CU is bidirectional and occurs three times in total. The operations highlighted in red are processed in parallel including low-level parallelization among the PEs (for weight updating, estimate calculating, resampling and state prediction) and high-level parallelization between the CU and PEs (while the PEs process updating, the CU performs resampling for the $z = z_0$ part; while the PEs process resampling, the CU selects the desired estimate and extracts them).

The parallelized SMC-PHD filter comprises four parts: state prediction, partial updating, MEAP estimate extraction and partial resampling. To simplify the description, we will present the weight updating, estimate extraction and resampling steps first, followed by the prediction step. The steps for one filtering iteration are described next:

Step 1. *Allocation*: Given that all underlying particles and observations are ready for use in the CU, then the CU distributes separate observations to different PEs and broadcasts all particles to each PE.

The following **Step 2–5** are processed in PEs in parallel (the low level parallelization) while the high-level parallelization is realized between PEs and the CU.

Step 2. *Weight updating*: Each PE processes a set of observations z in parallel to obtain $c_k(z,j)$, $C_k(z)$, $w_k(z,j)$ and $W_k(z)$. (Concurrently, the CU will resample according to $w_k(z_0,j)$ to obtain $\lceil M_p \times W_k(z_0) \rceil$ particles, where M_p is a specified parameter; see Section 3.2.3).

Step 3. *Estimate extraction*: Each PE calculates its estimates according to its observations based on $c_k(z,j)$, $p_D(x)$, as shown in (17). At the end of this step, PEs report $C_k(z)$, $W_k(z)$ and all the estimates $X_j, j = 1,\ldots,|Z_k|$ to the CU.

Step 4. *Resampling*: Resampling is performed on $w_k(z,j)$ with an expected number $\lceil W_k(z) \times M_p \rceil$ of particles to resample for each $z \in \{Z_k\}$ in PEs individually. The

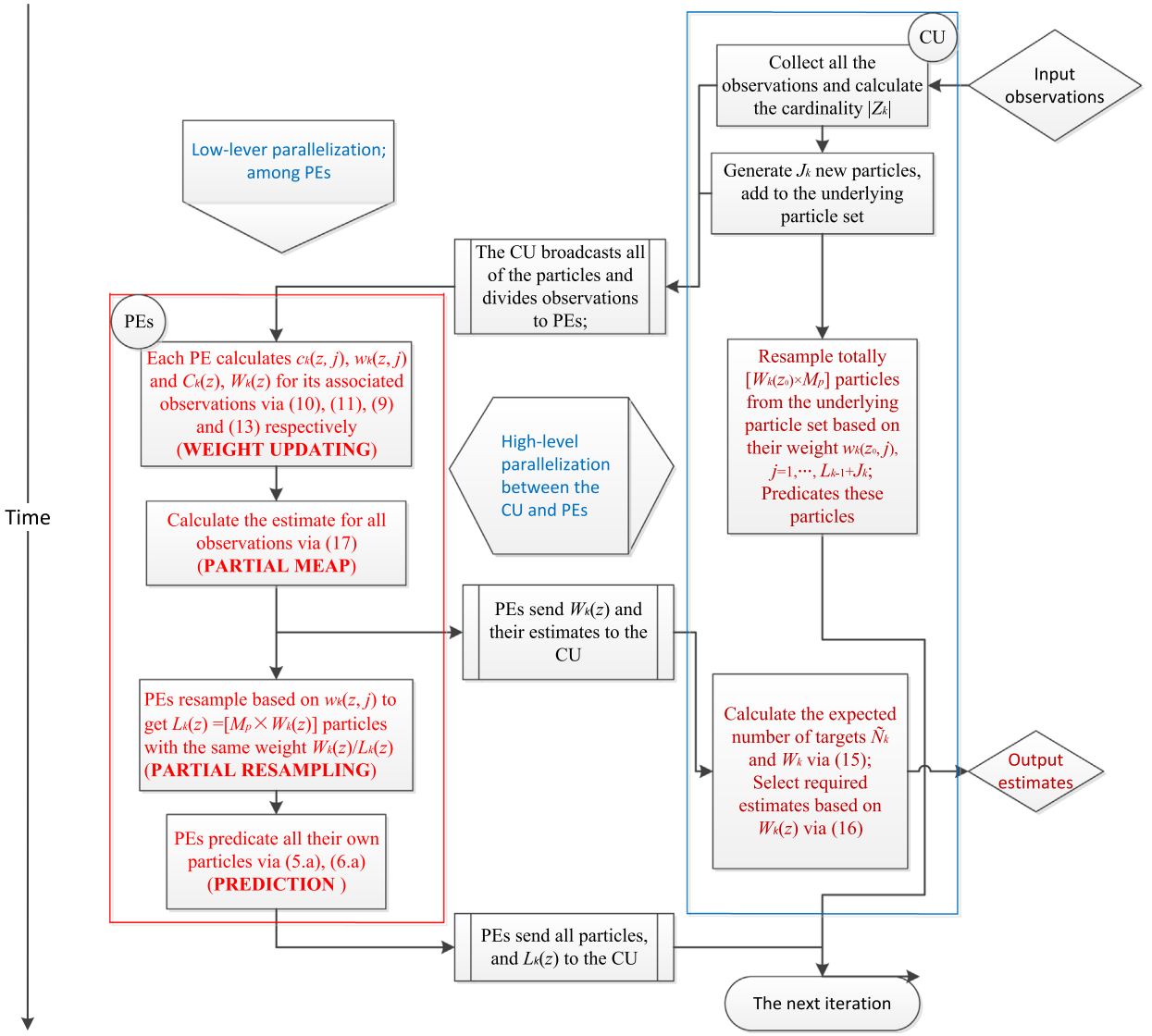

Fig. 2. The parallel processing framework of the SMC-PHD filter.

resampled particles are weighted the same as $(W_k(z)/[W_k(z)\times M_p])$

Step 5. *State Predication*: The resampled particles in each PE perform the state updating according to (5a). Then, PEs send all predicted states and $L_k(z)$ to the CU. The following step is performed in the CU:

Step 6. *Computation within CU (high-level parallelization)*: During the above steps 4 and 5 proceeded in PEs, the CU calculates the total weight mass $W_k=\sum_z W_k(z)$ and the estimated number of targets $\hat{N}_k=[W_k]$ and determines the required observations a_j, $j=1,\ldots,\min(\hat{N}_k,|Z_k|)$ based on $W_k(z)$ via (16) and correspondingly selects the corresponding estimates obtained in step 3. Also, the CU generates J_{k+1} new particles according to the birth target model (5b) and (6b). The predicted L_k particles reported from PEs in step 5 together with the new generated J_{k+1} particles form the new particle set in the CU that will be used for the next filtering iteration.

Then, all steps are repeated for the next iteration $k+1$ when new observations are available.

In summary, there are two levels of parallelization in our approach. The first is the low-level parallelization among PEs, in which the weight updating, resampling and estimate extraction are processed in parallel according to the weight components of particles, and the prediction step is implemented in parallel by dividing particles. The higher level parallelization refers to concurrent processing of PEs and the

CU. Most of the computations are parallelized at the first level with regard to observations, which is the primary part of our parallelization. In addition to these parallelization steps, the communication between PEs and the CU can be parallelized as well, e.g. the *allocation* operation given in step 1 is handled in a broadcasting manner.

The following sub-sections provide further analysis with respect to the four parts of the SMC-PHD filter.

3.2. Primary parallel processing

3.2.1. Partial weight updating

Remark 1. Resampling and MEAP are executed on the weight components in each PE. The integrated weight of each particle shown in (8) is not used in the filter and therefore does not need to be calculated.

Remark 2. The weight updating of particles and MEE are the primary computational steps of the filter. The computational load in each PE is proportional to the number of observations allocated. The number of observations allocated to different PEs is approximately even-balanced and the difference is no larger than one. Therefore it is fair to say that the computational load distributed to different PEs is approximately even-balanced and therefore, there is no need for a load balancing operation.

3.2.2. Partial MEAP estimate extraction

As shown in Algorithm 1, the Multi-EAP estimate extraction method consists of two steps: rank $W_k(z)$ for all $z \in Z_k$ to identify the largest $\min\left(\hat{N}_k, |Z_k|\right)$, and calculate their corresponding EAP estimate $s\ x_1^{EAP}, \ldots, x_{\min(N_k,|Z_k|)}^{EAP}$.

Determining Eq. (16) requires communication between PEs in the first step if it is directly applied even the NNN association is removed. To avoid this in our parallel processing approach, a slight change is made on the order of the two steps in calculating MEAP. Firstly, Eq. (17) is directly processed in PEs with respect to all observations $z_k \in Z_k$ after the updating step. Then, for estimate extraction, only the estimates corresponding to significant $W_k(z)$ are selected to output via Eq. (16) in the CU. Obviously, the calculation of (17) is approximately even-balanced among PEs. To note, if the NNN association is applied in the MEAP (that is particularly beneficial in the case of close-distributed targets [29,30]), it can be performed in the CU before distributing observations into PEs. Then, in addition to the particles and observations, their association information is also needed to be sent to PEs; we omitted the details here.

3.2.3. Partial resampling

Except the resampling step, all the other calculations are exactly the same between our parallelization and the serial implementation. Therefore, the unbiasedness of the resampling lies at the core of the unbiasedness of our parallelization and we will use exactly the same "unbiasedness" principle as serial resampling scheme [32].

Note that one advantage of the SMC-PHD filter over the basic particle filter is that the weights of particles do not need to be normalized for resampling. Normalization is less suitable for parallelization, which fortunately can be avoided in our approach. The weight mass of particles in a region corresponds to the expected number of targets in that region. Based on this, we resample in parallel according to the weight components of particles $w_k(z,j)$, $j = 1,2,\ldots, L_{k-1}+J_k$, $z \in Z_k$ in each PE with the same principle as the sequential procedure according to the integral weight. For $z = z_0$ the resampling is processed in the CU for the high-level parallelization, independent of the processing in PEs. Any unbiased sequential resampling method can be applied (e.g. the fast systematic scheme). The key is to determine the number of particles (that have uniform weight) to resample for each component in order to satisfy the unbiasedness condition. Since the weight mass of all particles represents the expected number of particles, the weight mass $W_k(z)$ has a clear meaning of the likelihood that the observation z is from a target. It is desirable to allocate a proportional number of particles to the expected number of targets, namely the weight mass $W_k(z)$. As done in the serial implementation, M_p particles are allocated for each expected target. For the weight component $w_k(z,j)$, $j = 1,2,\ldots, L_{k-1}+J_k$, a total of $L_k(z) = [W_k(z) \times M_p]$ particles are assigned to resample in our approach. In particular, the unbiasedness property is preserved by performing the rounding operation:

$$L_k(z) = \mathrm{E}\left(W_k(z) \times M_p\right), \quad z \in \{z_0, Z_k\} \tag{18}$$

In what follows, it is explicitly proved that the output of our component-based sampling approach is equivalent to the traditional method that is based on the entire/integral weight. The partial resampling scheme shares a similar idea with the stratified resampling [19] and if performed in serial manner, it can be taken as a type of unbiased compound resampling [32].

Theorem 2. *As long as condition (18) is satisfied, the parallel resampling will obtain the same unbiased result as the serial resampling based on the integrated weight, namely the number of times $L_k^{(j)}$ that each particle is resampled satisfies*

$$\mathrm{E}\left(L_k^{(j)} | w_{k|k}^{(j)}\right) = L_k \times \frac{w_{k|k}^{(j)}}{\sum_j^{L_{k-1}+J_k} w_{k|k}^{(j)}} \tag{19}$$

where $L_k^{(j)}$ is the number of times that the jth particle is resampled, L_k is the final number of resampled particles:

$$L_k = \sum_{z \in \{z_0, Z_k\}} L_k(z) \tag{20}$$

Proof. The unbiasedness condition is satisfied in parallel resampling in each PE for each z, i.e.

$$\mathrm{E}\left(L_k^{(j)}(z)\right) = L_k(z) \times \frac{w_k(z,j)}{W_k(z)} \tag{21}$$

where $L_k^{(j)}(z)$ denotes the number of times the jth particle is resampled with respect to observation z. As such, the total resampled number of the jth particle is

$$L_k^{(j)} = \sum_{z \in \{z_0, Z_k\}} L_k^{(j)}(z) \tag{22}$$

Based on the above premises, Eq. (19) is derived as follows:

$$
\begin{aligned}
& L_k \times \frac{w_{k|k}^{(j)}}{\sum_{j=1}^{L_{k-1}+J_k} w_{k|k}^{(j)}} \\
&= \sum_{z \in \{z_0,Z_k\}} L_k(z) \times \frac{\sum_{z \in \{z_0,Z_k\}} w_k(z,j)}{\sum_{z \in \{z_0,Z_k\}} W_k(z)} \quad \text{via(12), (13) and (20)} \\
&= \sum_{z \in \{z_0,Z_k\}} w_k(z,j) \times \frac{\sum_{z \in \{z_0,Z_k\}} L_k(z)}{\sum_{z \in \{z_0,Z_k\}} W_k(z)} \\
&= \sum_{z \in \{z_0,Z_k\}} w_k(z,j) \times \frac{\mathrm{E}\left(\sum_{z \in \{z_0,Z_k\}} W_k(z) \times M_p\right)}{\sum_{z \in \{z_0,Z_k\}} W_k(z)} \quad \text{via (18)} \\
&= \mathrm{E}\left(\sum_{z \in \{z_0,Z_k\}} w_k(z,j) \times M_p\right) \\
&= \mathrm{E}\left(\sum_{z \in \{z_0,Z_k\}} \left(w_k(z,j) \times M_p\right)\right) \\
&= \mathrm{E}\left(\sum_{z \in \{z_0,Z_k\}} \left(w_k(z,j) \times \frac{L_k(z)}{W_k(z)}\right)\right) \\
&= \mathrm{E}\left(\sum_{z \in \{z_0,Z_k\}} L_k^{(j)}(z)\right) \quad \text{via (21)} \\
&= E\left(L_k^{(j)} | w_{k|k}^{(j)}\right)
\end{aligned}
$$

3.2.4. Prediction

After resampling, each PE (as well as the CU) will directly update the state of the resampled particles based on the dynamic model (5a). These predicted particles are then sent to the CU. Concurrently, new weighted particles are generated for possible new-appearing targets in the CU based on (5b). Here, exchanging particles between PEs can be useful to balance the number of resampled particles among PEs, which will cause additional communication requirements. However, we do not suggest doing so since the prediction step is relatively computationally inexpensive, especially when a simple proposal function is adopted.

As an option, if the obtained number of particles is too small (e.g. lower than a minimum threshold), it can be self-resampled in the CU in an unbiased manner to supplement a specified number of particles.

4. Qualitative comparison of parallel and serial implementation

Distributing tasks into PEs for parallel processing will introduce additional communication overhead between PEs and the CU. The overall computing speed can only be increased by parallelization when the time of parallel computation, which includes communication overhead, is smaller than sequential execution time. To quantitatively study the parallel efficiency, we denote the following symbols for computational operations (with regard to Fig. 2):

$S_{i,CU}$: Time needed for the CU to broadcast all particles and divide separate observations to PE i
$S_{i,PE}$: Time needed for PE i to send all particles, $L_k(z)$ and $W_k(z)$ to the CU
$S_{i,CW}$: Time needed for PE i to report $C_k(z)$ and $W_k(z)$ to the CU
$S_i \triangleq S_{i,CU}+S_{i,PE}+S_{i,CW}$
$PE_{i,pre}$: Time used for prediction at PE i
$PE_{i,upd}$: Time used for updating at PE i
$PE_{i,res}$: Time used for resampling at PE i
$PE_{i,MEAP}$: Time used for MEAP and sending to CU at PE i
$PE_i \triangleq PE_{i,pre}+PE_{i,upd}+PE_{i,res}+PE_{i,MEAP}$
CU_J: Time needed for the CU to generate J_k new particles
CU_Z: Time needed for the CU to divide Z_k and calculate $|Z_k|$
CU_{res}: Time needed for the CU to resample according to z_0
CU_{est}: Time needed for the CU to extract estimates
$CU_{Optional}$: Time needed for the CU to calculate L_k, and to supplement particles via self-resampling if L_k is too small
$CU \triangleq CU_J+CU_Z+CU_{res}+CU_{est}+CU_{optional}$

We introduce the following assumptions in our model:

(**a.1**) communication between PEs and CU can be processed one by one and only in one direction at the same time;
(**a.2**) all PEs have the same processing capability and for the same task, the CU will spend λ times more or less time as compared with one PE.
(**a.3**) all PEs and the CU are maximally utilized and will not stop until no task is left to be done.

In the serial computing realised on the CU, there is no communication required. The computing time required by the particle PHD filter (including estimate extraction) can be written as

$$T_{serial} = \sum_{i=1}^{p} \lambda PE_i + CU \tag{23}$$

The computing time required by our parallel implementation of the SMC-PHD filter satisfies

$$T_{parallel} \le \max_i (PE_i) + \sum_{i}^{P} S_i + CU \tag{24}$$

As analysed in Remark 2, the most computationally intensive operations of weight updating and MEE are naturally well balanced among PEs. Taking into account assumptions **a.2–3**, all PEs have the same computing capabilities and a very similar computational load, resulting in $PE_i \approx PE_j$, $S_i \approx S_j$ for any i,j. Then, (24) can be approximated as

$$T_{parallel} \le PE_i + P \times S_i + CU \tag{25}$$

The filter can benefit from the parallelization as long as $T_{parallel} < T_{serial}$. Comparing (23) with (25), we can easily obtain a sufficient condition to satisfy $T_{parallel} < T_{serial}$ as

$$PE_i/P + S_i < \lambda PE_i \tag{26}$$

If the computing capability of the PE is equivalent to the CU, then, (26) will be simplified into

$$S_i < (1-1/P)PE_i \tag{27}$$

For $P=2$ (two PEs are used), (27) becomes $S_i<0.5PE_i$, for $P=4$, (27) becomes $S_i<0.75PE_i$ and so on.

Remark 3. Condition (27) indicates that more PEs (fully used), smaller S_i and larger PE_i all indicate better parallelization speedup. This is the reason our approach tries to reduce the communication between PEs and the CU and to parallelize all calculations to the largest degree.

5. Quantitative experiments

The experiments are based on shared-memory multi-core architecture computers, where communication among cores is accomplished through read and write access to the shared memory. In a simple albeit classic two-dimensional multi-target tracking scenario over the region $[-100,100]\times[-100,100]$, new targets are assumed to appear according to a Poisson point process with intensity function $\gamma_k=0.2N(.;\overline{x},Q)$, where $\overline{x}=[0,3,0,-3]^T$, $Q=\mathrm{diag}([10,1,10,1]^T)$, and diag (a) represents a diagonal matrix with diagonal a. The Markov transition equation that characterizes the nearly constant velocity target dynamics is given as

$$x_k=\begin{bmatrix}1&t&0&0\\0&1&0&0\\0&0&1&t\\0&0&0&1\end{bmatrix}x_{k-1}+\begin{bmatrix}t^2/2&0\\t&0\\0&t^2/2\\0&t\end{bmatrix}\begin{bmatrix}v_{k,1}\\v_{k,2}\end{bmatrix} \tag{28}$$

where the sampling time $t=1$, $x_k=[x_{k,1},x_{k,2},x_{k,3},x_{k,4}]^T$, $[x_{k,1},\ x_{k,3}]^T$ is the position while $[x_{k,2},\ x_{k,4}]^T$ is the velocity at time k. The process noise $\{v_{k,1}\}$, $\{v_{k,2}\}$ are mutually independent zero-mean Gaussian white noise with a respective standard deviation of 1 and 0.1.

The Cartesian position observation equation is given by

$$z_k=\begin{bmatrix}1&0&0&0\\0&0&1&0\end{bmatrix}x_k+\begin{bmatrix}u_{k,1}\\u_{k,2}\end{bmatrix} \tag{29}$$

with $\{u_{k,1}\}$ and $\{u_{k,2}\}$ as mutually independent zero-mean Gaussian white noise with the same standard deviation of 2.5. Clutter is uniformly distributed over the region with an average rate of r points per scan, i.e. $\kappa=(r/200^2)$.

Without loss of generality, targets may survive for a long or short period of time, appear jointly, adjacently or solitarily, and their trajectories may cross each other, be far away or close as shown in Figs. 3 and 4. The survival probability and the detection probability of each target are set as $p_S=0.95$, $p_D=0.95$ respectively. Fixed M_p particles per expected target are used in resampling to adjust the number of particles. Both the serial and parallel SMC-PHD filters employ the MEAP estimator (free of the NNN association) for computing fast and accurate estimate extraction.

Fig. 5 gives the optimal sub-pattern assignment (OSPA) metric [40] result (the mean estimate and the standard deviation over 100 Monte Carlo trials) of both serial and parallel filters (using 2 PEs) in our first experiment for $r=10$, $M_p=500$. The output results of both filters are indeed very close and the difference is less than 0.6% on average of 100 trials (this slight difference is due to the random number as the random numbers generated each time are different). This agrees with the theoretical study that our parallelization provides the same estimation result as the serial implementation. In the following experiments, we put the emphasis on the computing efficiency of the parallelization.

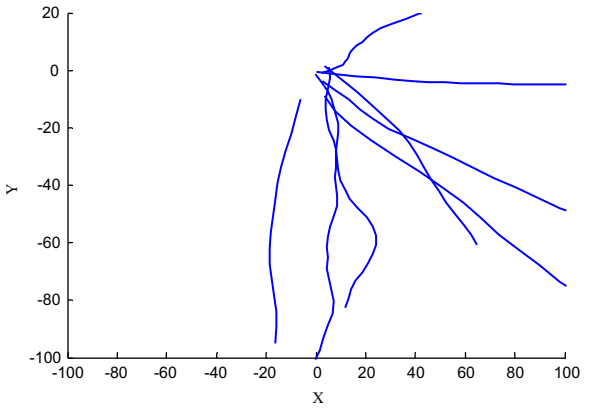

Fig. 3. Trajectories of targets in the scene.

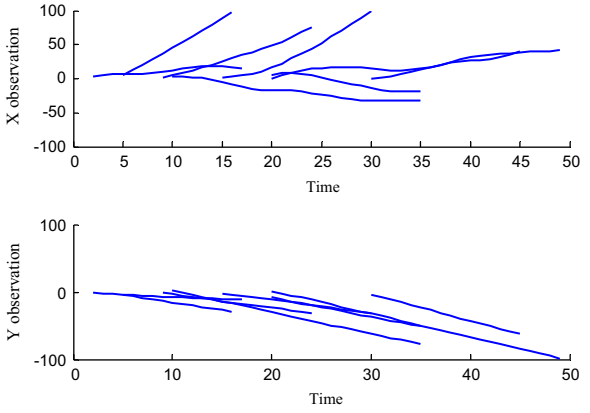

Fig. 4. Trajectories of targets in X and Y dimensions separately.

The computational load required by the SMC-PHD filter is in proportion to the number of observations and the number of particles, which depend on the parameters r and M_p. Without loss of generality, different r and M_p are explored in a large range to compare the computing speed of the proposed parallel approach with the serial one. Two experiment platforms are used respectively and in both, we first use different degree of clutter, i.e. r is set from 1 to 50 with interval 1, in which $M_p=500$. We then use different M_p from 200 to 2000 with interval 50, in which $r=10$; here the observations are the same for all trials.

5.1. Parallel computation with MATLAB

We point out that the presented algorithm is not hardware specified and can be implemented on different parallel processing platforms. However, we programme on the

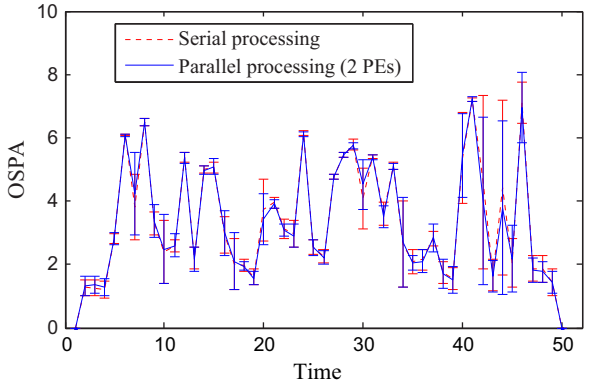

Fig. 5. Mean OSPA and its variance over 100 trials.

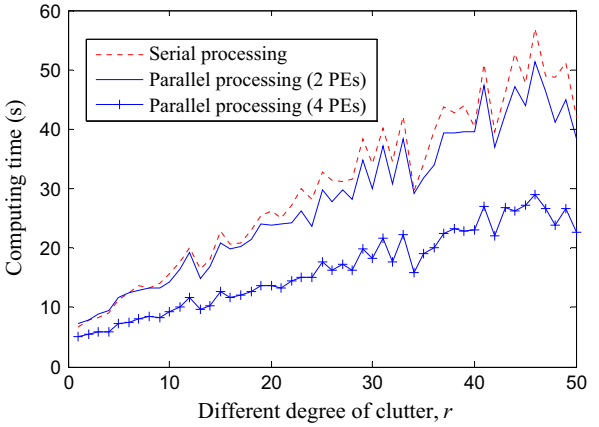

Fig. 6. Average computing time against different degrees of clutter ($M_p = 500$).

MATLAB® platform because it is friendly and popular in the signal processing community, which provides two main approaches for parallel computation. One is the built-in multithreading and the other is the multi-core/processor computation based on the Parallel Computing Toolbox, which deals with each thread as a separate 'worker' (corresponding to the PE in our algorithm). Maximally 4 PEs are available for the first experiment platform with i5-3450 CPU while, maximally 8 PEs are available for the second experiment platform with Intel Xeon(R) CPU E5540. To capture the average performance, we run 10 trials for each experiment with the same target trajectories and observations.

5.2. Multi-core computing

This experiment is executed on a computer with Intel i5-3450 CPU that consists of a total of 4 cores and 4 threads with a clock frequency of 3.10 GHz. The operating system is Windows 7. In this experiment, two and four PEs are used separately for parallel processing. Fig. 6 shows average computing time for different degrees of clutter r and Fig. 7 shows the average computing time for different numbers of particles M_p per target. The computing speed increased by parallelization on average as compared to the serial implementation for some points is summarized in Table 1.

The average computing time of each step is given in Fig. 8 for $r = 0$, $M_p = 500$ and in Fig. 9 for $r = 10$, $M_p = 500$. As shown in Fig. 4, there is no target for $k \in \{1, 50\}$, only one target for $k \in \{2\text{–}4, 46\text{–}49\}$, two targets for $k \in \{5\text{–}8, 36\text{–}45\}$ and more targets for $k \in \{9\text{–}40\}$. The number of targets contributes to the computing time twofold: first, the more targets, the more particles; second, the more targets, the more observations. The computing time increases with both the number of particles used and the number of observations to handle. Our parallel approach aims to alleviate the latter (reducing the increase of the computing time with the number of observations) but not the former. In particular, when the number of targets is low (≤ 2) and $r = 0$, the number of observations is relatively small and parallelization does not speed up but slows down the processing as compared

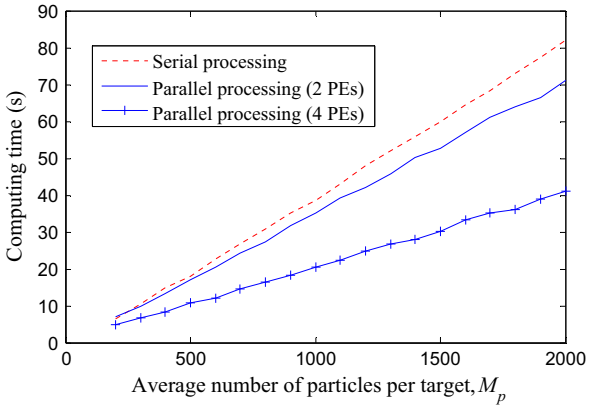

Fig. 7. Average computing time against average number of particles allocated per target ($r = 10$).

Table 1
Speedup obtained by parallelization versus serial processing (%)

Condition	$r=5$, $M_p=1000$	$r=50$, $M_p=1000$	$r=10$, $M_p=500$	$r=10$, $M_p=2000$
2 PEs	−2.5	8.6	5.3	13.2
4 PEs	36.0	45.8	27.5	49.7

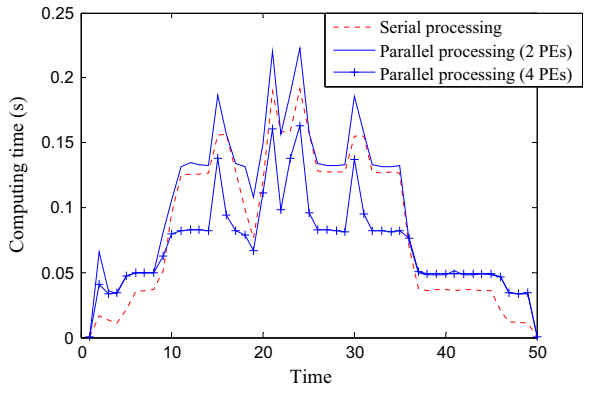

Fig. 8. Average computing time of each step ($r = 0$, $M_p = 500$).

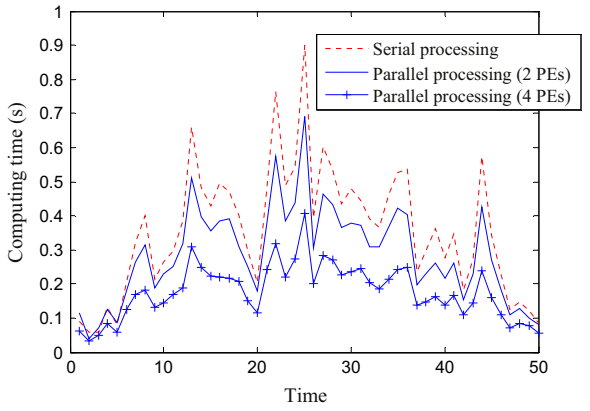

Fig. 9. Average computing time of each step ($r=10$, $M_p=500$).

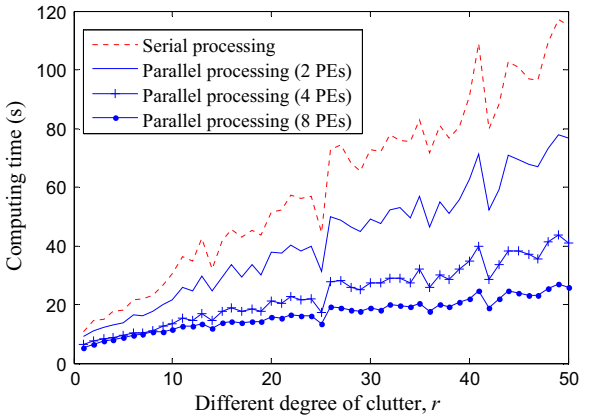

Fig. 10. Average computing time against different degree of clutter ($M_p=500$).

to the serial filter shown in Fig. 8. But if $r=10$ (the clutter generates on average 10 observations per scan), our parallelization is beneficial even when the number of targets is small, as shown in Fig. 9. These results indicate that our parallelization can benefit in cases of dense observations, regardless of whether they are generated by targets or by clutter. For the scene of relatively small number of targets and fewer clutter our parallelization might not be beneficial.

The results show that, the parallelization by using two processors has not improved the computing speed much (less than 14%). This indicates $S_i \approx 0.5PE_i$ for condition (27). In contrast, the parallelization by using four processors has significantly improved the computing speed (from 27% to 50%). When the number of clutter and particles used is large, the implementation can especially benefit from parallelization as indicated by Remark 3.

5.3. Multithreading computing

The second experiment is executed on a computer with Intel Xeon(R) CPU E5540 consisting of a total of 4 cores running at 2.53 GHz. Maximally 8 threads can be exploited in this system. The operating system is Linux. In this experiment, two PEs, four PEs and eight PEs are used separately for parallel processing. The computing times consumed by the serial SMC-PHD filter and the parallel SMC-PHD filter are given in Fig. 10 for different r and in Fig. 11 for different numbers of particles used per target M_p. The increase in computing speed with parallelization as compared to the serial implementation for some conditions is summarized in Table 2.

The results show that the parallelization is beneficial and that it scales well with the increasing number of PEs used. Particularly, when the number of clutter and the number of particles used are both large, parallelization is the most beneficial. In contrast, when the number of clutter and the number of particles used are both small, there is less benefit from parallelization and using more PEs is not suggested. This is simply because, with the increase of number of PEs, the resource utilization and the communication overhead for parallel processing increase

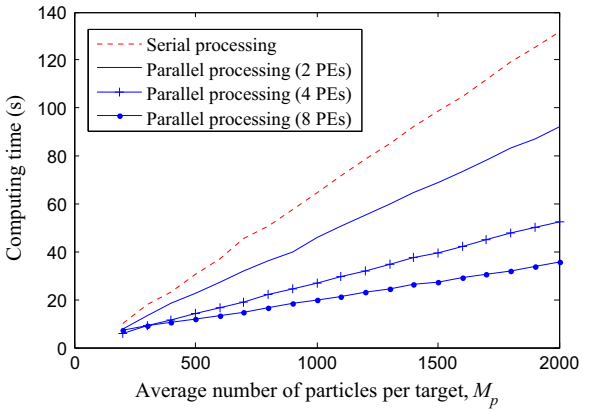

Fig. 11. Average computing time against average number of particles allocated per target ($r=10$).

as well. For a certain level of the number of targets and clutter, there is a trade-off between parallelization level and the communication overhead.

6. Conclusion

An algorithmic parallel-processing framework for the popular SMC-PHD filter is proposed, based on the insightful understanding of the filter. To achieve high-level of parallelism in each calculation of the SMC-PHD filter, we propose efficient solutions separately for all the four steps of the filter including state prediction, weight updating, resampling and multi-estimate extraction. The state prediction step is parallelized in terms of distributing particles; the weight updating, resampling and estimate extraction steps are parallelized in terms of distributing measurements. This combination of two different distributing solutions is based on the unbiased resampling, which is critical for obtaining unbiased estimates and achieving full parallelization. On average, the computational load is approximately equivalent among processors. More importantly, the proposed parallel implementation provides the same result as the serial implementation.

Table 2
Speedup obtained by parallelization versus serial processing (%)

Condition	$r=5, M_p=1000$	$r=50,\ M_p=1000$	$r=10, M_p=500$	$r=10,\ M_p=2000$
2 PEs	22.9	33.3	26.0	30.1
4 PEs	46.7	64.3	53.8	60.3
8 PEs	52.3	77.7	60.7	73.1

Experiments have confirmed the validity of our approach which has gained considerable parallelization speedup. Since the software of Matlab does not give enough control to the programmer, we expect that better speedup can be obtained by using parallel hardware and software that allow for lower-level parallel programming.

Acknowledgement

This work is partly supported by Ministry of Economy and Finance of Spain (ref. TIN2012-36586-C03-03), EU FP7 (ref. PIRSES-GA-2012-318878) and National Natural Science Foundation of China (ref. 51475383). Tiancheng Li's work is supported by the Excellent Doctorate Foundation of Northwestern Polytechnical University and the Postdoctoral Fellowship of the University of Salamanca.

References

[1] G.W. Pulford, Taxonomy of multiple target tracking methods, IEE Proc. Radar Sonar Navig. 152 (5) (2005) 291–304.
[2] R. Mahler, Multi-target Bayes filtering via first-order multi-target moments, IEEE Trans. Aerosp. Electron. Syst. 39 (4) (2003) 1152–1178.
[3] R. Streit, Multisensor multitarget intensity filter, in: Proceedings of the 11th International Conference on Information Fusion, Cologne, Germany, June 30–July 3, 2008.
[4] P. Braca, S. Marano, V. Matta, P. Willett, Asymptotic efficiency of the PHD in multitarget/multisensor estimation, IEEE J. Sel. Top. Signal Process. 7 (3) (2013) 553–564.
[5] M. Pace, P. Del Moral, Mean-field PHD filters based on generalized Feynman–Kac flow, IEEE J. Sel. Top. Signal Process. 7 (3) (2013) 484–495.
[6] O. Erdinc, P. Willett, Y. Bar-Shalom, The bin-occupancy filter and its connection to the PHD filters, IEEE Trans. Signal Process. 57 (11) (2009) 4232–4246.
[7] M. Uney, B. Mulgrew, D. Clark, Cooperative sensor localisation in distributed fusion networks by exploiting non-cooperative targets, in: Proceedings of the 2014 IEEE Workshop on Statistical Signal Processing, Gold Coast, Australia, 29 June–2 July 2014.
[8] B. Ristic, D.E. Clark, N. Gordon, Calibration of multi-target tracking algorithms using non-cooperative targets, IEEE J. Sel. Top. Signal Process. 7 (3) (2013) 390–398.
[9] J. Mullane, B. Vo, M.D. Adams, W.S. Wijesoma, A random set formulation for Bayesian SLAM, in: Proceedings of the 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, Nice, France, 22–26 September 2008.
[10] B.N. Vo, S. Singh, A. Doucet, Sequential Monte Carlo methods for multi-target filtering with random finite sets, IEEE Trans. Aerosp. Electron. Syst. 41 (4) (2005) 1224–1245.
[11] B.N. Vo, W.K. Ma, The Gaussian mixture probability hypothesis density filter, IEEE Trans. Signal Process. 54 (11) (2006) 4091–4104.
[12] D. Clark, B.T. Vo, B.N. Vo, Gaussian particle implementations of probability hypothesis density filters, in: Proceedings of the 2007 IEEE Aerospace Conference, Big Sky, Montana, March 2007.
[13] M. Schikora, A. Gning, L. Mihaylova, D. Cremers, W. Koch, Box-particle probability hypothesis density filtering, IEEE Trans. Aerosp. Electron. Syst. 50 (3) (2014) 1660–1672.
[14] B. Ristic, B.T. Vo, B.N. Vo, A. Farina, A tutorial on Bernoulli filters: theory, implementation and applications, IEEE Trans. Signal Process. 61 (13) (2013) 3406–3430.
[15] R. Mahler, PHD filters of higher order in target number, IEEE Trans. Aerosp. Electron.Syst. 43 (4) (2007) 1523–1543.
[16] C. Ouyang, H.B. Ji, Z.Q. Guo, Extensions of the SMC-PHD filters for jump Markov systems, Signal Process. 92 (6) (2012) 1422–1430.
[17] B.K. Habtemariam, R. Tharmarasa, T. Kirubarajan, PHD filter based track-before-detect for MIMO radars, Signal Process. 92 (3) (2012) 667–678.
[18] T. Li, S. Sun, T.P. Sattar, High-speed sigma-gating SMC-PHD filter, Signal Process. 93 (9) (2013) 2586–2593.
[19] T.M. Wood, D. Clark, B. Ristic, Efficient resampling and basic track continuity for the SMC-PHD filter, in: Proceedings of Cognitive Systems with Interactive Sensors, Crawley, UK, 2010.
[20] S. Hong, L. Wang, Z. Shi, K. Chen, Simplified particle PHD filters for multiple target tracking: algorithm and architecture, Prog. Electromagn. Res. 120 (2011) 481–498.
[21] Z. Shi, Y. Zheng, X. Bian, Z. Yu, Threshold-based resampling for high-speed particle PHD filter, Prog. Electromagn. Res. 136 (2013) 369–383.
[22] Z. Shi, Y. Liu, S. Hong, J. Chen, X. Shen, POSE: design of hardware-friendly particle-based observation selection PHD filter, IEEE Trans. Ind. Electron. 61 (4) (2014) 1944–1956.
[23] M. Del Coco, A. Cavallaro, Parallel particle-PHD filter, in: Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Florence, Italy, 4–9 May 2014.
[24] M. Bolić, P.M. Djurić, S. Hong, Resampling algorithms and architectures for distributed particle filters, IEEE Trans. Signal Process. 53 (7) (2005) 2442–2450.
[25] S.J. Hong, P.M. Djurić, High-throughput scalable parallel resampling mechanism for effective redistribution of particles, IEEE Trans. Signal Process. 54 (3) (2006) 1144–1155.
[26] L. Zhao, P. Ma, X. Su, H. Zhang, A new multi-target state estimation algorithm for PHD particle filter, in: Proceedings of the 13th International Conference on Information Fusion, Edinburgh, UK, 2010.
[27] B. Ristic, D. Clark, B.N. Vo, Improved SMC implementation of the PHD filter, in: Proceedings of the 13th International Conference on Information Fusion, Edinburgh, UK, 2010.
[28] M. Schikora, W. Koch, R. Streit, D. Cremers, Sequential Monte Carlo method for multi-target tracking with the intensity filter, in: P. Georgieva, L. Mihaylova, L.C. Jain (Eds.), Springer, 2012, pp. 55–87.
[29] T. Li, J.M. Corchado, S. Sun, H. Fan, Multi-EAP: extended EAP for multi-estimate extraction for the SMC-PHD filter, IEEE Trans. Aerosp. Electron. Syst. (2015). (submitted for publication).
[30] T. Li, S. Sun, J.M. Corchado, M.F. Siyau, A particle dyeing approach for track continuity for the SMC-PHD filter, in: Proceedings of the 17th International Conference on Information Fusion, Salamanca, Spain, July 7–10, 2014.
[31] K. Punithakumar, T. Kirubarajan, A. Sinha, Multiple-model probability hypothesis density filter for tracking maneuvering targets, IEEE Trans. Aerosp. Electron. Syst. 44 (1) (2008) 87–98.
[32] T. Li, M. Bolic, P. Djuric, Resampling methods for particle filtering: classification, implementation, and strategies, IEEE Signal Process. Mag. 32 (3) (2015) 70–86.
[33] M. Üney, D.E. Clark, S.J. Julier, Distributed fusion of PHD filters via exponential mixture densities, IEEE J. Sel. Top. Signal Process. 7 (3) (2013) 521–534.
[34] G. Battistelli, L. Chisci, C. Fantacci, A. Farina, A. Graziano, Consensus CPHD filter for distributed multitarget tracking, IEEE J. Sel. Top. Signal Process. 7 (3) (2013) 508–520.
[35] B.K. Habtemariam, A. Aravinthan, R. Tharmarasa, K. Punithakumar, T. Lang, T. Kirubarajan, Distributed tracking with a PHD filter using efficient measurement encoding, J. Adv. Inf. Fusion 7 (2) (2012) 1–17.

[36] O. Hlinka, F. Hlawatsch, P.M. Djurić, Consensus-based distributed particle filtering with distributed proposal adaptation, IEEE Trans. Signal Process. 62 (12) (2014) 3029–3041.

[37] O. Hlinka, F. Hlawatsch, P.M. Djurić, Distributed particle filtering in agent networks, IEEE Signal Process. Mag. 30 (1) (2013) 61–81.

[38] C.S. Lee, J. Franco, J. Houssineau, D. Clark, Accelerating the single cluster PHD filter with a GPU implementation, in: Proceedings of the 2014 International Conference on Control, Automation and Information Sciences, Melbourne, Australia, 11–12 December, 2014.

[39] T. Chen, T.B. Schön, H. Ohlsson, L. Ljung, Decentralized particle filter with arbitrary state decomposition, IEEE Trans. Signal Process. 59 (2) (2011) 465–478.

[40] D. Schuhmacher, B.T. Vo, B.N. Vo, A consistent metric for performance evaluation in multi-object filtering, IEEE Trans. Signal Process. 56 (8) (2008) 3447–3457.