PARALLEL PARTICLE-PHD FILTER

Marco Del Coco

Innovation Engineering Department
University of Salento, Italy

Andrea Cavallaro

Centre for Intelligent Sensing
Queen Mary University of London, UK

ABSTRACT

The complexity of multi-target tracking grows faster than linearly with the increase of the numbers of objects, thus making the design of real-time trackers a challenging task for scenarios with a large number of targets. The Probability Hypothesis Density (PHD) filter is known to help reducing this complexity. However, this reduction may not suffice in critical situations when the number of targets, dimension of the state vector, clutter conditions and sample rate are high. To address this problem, we propose a parallelization scheme for the particle PHD filter. The proposed scheme exploits the knowledge of mutual interacting targets in the scene to help fragmentation and to reduce the workload of individual processors. We compare the proposed approach with alternative parallelization schemes and discuss its advantages and limitations using the results obtained on two multi-target tracking datasets.

Index Terms— Multi-target tracking, PHD filter, parallelism.

1. INTRODUCTION

Multi-target tracking (MTT) is a widely investigated problem that requires dealing with concurrent and potentially interacting targets [1–3]. Although Monte Carlo methods [4] (particle filters) are widely used for target tracking, the number of particles grows exponentially with the number of targets, potentially leading to computationally intractable multi-target states. The Random Finite Set (RFS) [5] theory offers an effective framework for MTT and, coupled with the Bayes recursion, can account for several tracking challenges, such as the growing state dimension for multi-target problems, false or missing measurements, target births and deaths. This framework is represented by the Probability Hypothesis Density filter (PHD) that propagates the first-order moment of the multi-target posterior instead of the full posterior itself [6]. The PHD filter can be implemented as a particle filter (P-PHD) [3, 7] and time-efficiency can be addressed with a pipeline approach [8]. However, this approach is limited by the number of pipeline levels and does not take advantage of current multi-processors systems. An appealing solution is that of parallel computing that, to the best of our knowledge, has not been yet investigated for the P-PHD.

Naive parallelizations distribute particles among processors [9], fragmenting the data over processors and being limited by data dependency constraints. The over-linearity relation between the amount of data to be analyzed and the computational load makes the perspective of an additional fragmentation more attractive.

This work was carried out while the first author was visiting the Centre for Intelligent Sensing, Queen Mary University of London. A. Cavallaro acknowledges the support of the Artemis JU and UK Technology Strategy Board as part of the Cognitive & Perceptive Cameras (COPCAMS) project under GA number 332913.

In this paper we propose a parallel algorithm for P-PHD that takes advantage of the knowledge of the particle location in the multi-target state. The observation of multi-target states is key to assume possible independences between groups of particles that are associated to targets that are distant from each other in the state space. These particles can be treated separately and spread among different processors. This solution results is a super-fragmentation that reduces the computational complexity. We compare the proposed solution with alternative parallelization approaches in order to quantify the improvements in performance and discuss its advantages and limitations.

This paper is organized as follows. Sec. 2 describes the P-PHD. Sec. 3 introduces the proposed parallelization approach based on the observation of scene behavior and possible sub-optimal adjustments. In Sec. 4 we analyze the performance of the proposed approach and compare it with alternative solutions. In Sec. 5 we draw the conclusions.

2. PARTICLE PHD FILTER

Let $\boldsymbol{x}_k$=$(x_k, \dot{x}_k, y_k, \dot{y}_k) \in E_s$ be a target state at current time k where (x_k, y_k) is the position and $(\dot{x}_k, \dot{y}_k)$ the velocity of the target, and E_s the state space. Let $\boldsymbol{z}_k$=$(x_{z_k}, y_{z_k}) \in E_O$ be the single-target observation and E_O the observation space. $\boldsymbol{X}_k$=$\{\boldsymbol{x}_{k,1}, \ldots, \boldsymbol{x}_{k,M(k)}\}$ and $\boldsymbol{Z}_k$=$\{\boldsymbol{z}_{k,1}, \ldots, \boldsymbol{z}_{k,N(k)}\}$ are the multi-target state and the set of observations at k, respectively, and $M(k)$ and $N(k)$ are the number of targets and observations.

P-PHD employs a Monte Carlo method to approximate the PHD with a set of weighted samples [3, 7]. Let $\{\omega_k^{(i)}, \boldsymbol{x}_k^{(i)}\}_{i=1}^{L_k}$ be the set of L_k weight-state (particles) pairs that approximate the multi-target state distribution at k, with $\omega_k^{(i)}$ being the weight of particle $\boldsymbol{x}_k^{(i)}$. The particles at $k-1$ are propagated according to a prior model to obtain a prediction at k. L_{k-1} particles at $k-1$ are *propagated* using the importance function $q_k(\cdot|\boldsymbol{x}_{k-1}^{(i)}, \boldsymbol{Z}_k)$. When measurements at k are available, J_k *new particles* are drawn from the importance function defined as $p_k(\cdot|\boldsymbol{Z}_k)$ [7]. An approximation of the predicted distribution of the multi-target state at k can now be expressed as $\{\tilde{\omega}_k^{(i)}, \tilde{\boldsymbol{x}}_k^{(i)}\}_{i=1}^{L_{k-1}+J_k}$. The values of the predicted weights $\tilde{\omega}_{k|k-1}^{(i)}$ can be computed as [7]:

$$\tilde{\omega}_{k|k-1}^{(i)} = \begin{cases} \frac{\phi_k(\tilde{\boldsymbol{x}}_k^{(i)}, \boldsymbol{x}_{k-1}^{(i)})\omega_{k-1}^{(i)}}{q_k(\tilde{\boldsymbol{x}}_k^{(i)}|\boldsymbol{x}_{k-1}^{(i)}, \boldsymbol{Z}_k)}, & i = 1, \ldots, L_{k-1} \\ \frac{\gamma_k(\tilde{\boldsymbol{x}}_k^{(i)})}{J_k p_k(\tilde{\boldsymbol{x}}_k^{(i)}|\boldsymbol{Z}_k)}, & i = L_{k-1}+1, \ldots, L_{k-1}+J_k. \end{cases} \quad (1)$$

where $\phi_{k|k-1}(x, \xi) = e_{k|k-1}(\xi) f_{k|k-1}(x|\xi)$ is the analogue of the single-target state transition probability, $e_{k|k-1}(\xi)$ is the probability that the target still exists at k and $\gamma_k(\cdot)$ is the intensity function of

new-born targets [7]. When measurements are available, weights are updated based on the observations.

Before the update and resampling steps, it is necessary to split the particle set over different *time-strata* so that particles can be grouped based on the time instant in which they were created. Next, the set $\{\tilde{\omega}_k^{(i)}, \tilde{\boldsymbol{x}}_k^{(i)}\}_{i=1}^{L_{k-1}+J_k}$ is decomposed into S sets $\{\tilde{\omega}_{k,s}^{(i)}, \tilde{\boldsymbol{x}}_{k,s}^{(i)}\}_{i=1}^{L_{k,s}}$ where s=1, ..., S is the time-stratum index. Therefore the update and resampling steps can be independently performed over different time-strata using the measurement set $\boldsymbol{Z}_k$.

The weights are updated using [7]:

$$\tilde{\omega}_k^{(i)} = \left[p_M(\tilde{\boldsymbol{x}}_k^{(i)}) + \sum_{\boldsymbol{z} \in \boldsymbol{Z}_k} \frac{\psi_{k,\boldsymbol{z}}(\tilde{\boldsymbol{x}}_k^{(i)})}{\kappa_k(\boldsymbol{z}) + C_k(\boldsymbol{z})} \right] \tilde{\omega}_{k|k-1}^{(i)}, \quad (2)$$

where $p_M(x)$ is the miss-detection probability, $\kappa_k(z)$ is the clutter intensity, $\psi_{k,\boldsymbol{z}}(x)$=$(1 - p_M(x))g_k(z|x)$, $g_k(z|x)$ is the likelihood that z is generated by a target with state x, and

$$C_k(\boldsymbol{z}) = \sum_{j=1}^{L_{k-1}+J_k} \psi_{k,\boldsymbol{z}}(\tilde{\boldsymbol{x}}_k^{(j)}) \tilde{\omega}_{k|k-1}^{(j)}. \quad (3)$$

The layer index s is omitted for simplicity of notation.

After the update step a *Multistage Multinomial Resampling* (MMR) is performed separately over different S strata [7]. MMR is necessary because a resampling over the whole particle set might delete new-born particles with low weights, which might be much smaller than the weights of older particles. For this reason, it is preferable to split particles into multiple time-strata and to perform a separate resampling over each stratum. Particles are then clustered to locate peaks of the multimodal distribution describing the multi-target state. Clustering generally operates only over a predefined number of the oldest time-strata to avoid the presence of particles survived to the last ones.

The state transition $f_{k|k-1}(\boldsymbol{x}_k|\boldsymbol{x}_{k-1})$ has generally a first order Gaussian dynamic [7], where the acceleration of each target is modeled as a random process with a fixed standard deviation:

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{A} & 0_2 \\ 0_2 & \boldsymbol{A} \end{bmatrix} \boldsymbol{x}_{k-1} + [n_p \; n_v \; n_p \; n_v]^T, \quad (4)$$

where $\boldsymbol{A}$=$\begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix}$, n_p and n_v are the independent white Gaussian noises over position and velocity with variance of σ_p and σ_v, respectively; dt is the sample time step, 0_2 is a 2 by 2 matrix of zeros and T is the transpose operator. The single-target likelihood can be defined as $g_k(\boldsymbol{z}|\boldsymbol{x})$=$\mathcal{N}(\boldsymbol{z}; \boldsymbol{H}\boldsymbol{x}, \Sigma(\boldsymbol{x}))$, where $\boldsymbol{H}$=$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$, $\Sigma(\boldsymbol{x})$ is a diagonal matrix and $\text{diag}(\Sigma(\boldsymbol{x})) = ([\sigma_p, \sigma_v, \sigma_p, \sigma_v])$. If knowledge of the scene is not available, we can assume $p_M(x)$, $e_{k|k-1}(\xi)$, $\gamma_k(x)$, $\kappa_k(x)$ to be constant over the scene [7].

New particles are generated from the density $p_k(\boldsymbol{x}_k|\boldsymbol{Z}_k)$ that is obtained as follows. For the target position, particles are spread around the measurements by

$$p_{p_k}([x_k, y_k]|\boldsymbol{Z}_k) = \frac{1}{N(k)} \sum_{\boldsymbol{z} \in \boldsymbol{Z}_k} \mathcal{N}(\boldsymbol{x}; [x_{z_k}, y_{z_k}], \Sigma_b(\boldsymbol{z})), \quad (5)$$

which is a mixture of Gaussians centered in $\boldsymbol{Z}$. $\Sigma_b(\boldsymbol{z})$ is a 4×4 diagonal matrix defined as $\text{diag}(\Sigma_b(\boldsymbol{z})) = [\sigma_{b_p}, \sigma_{b_v}, \sigma_{b_p}, \sigma_{b_v}]$. The target velocity component is instead modeled as $p_{v_{x_k}}(\hat{x}_k) = \mathcal{U}(v_{min_x}, v_{max_x})$ for x and $p_{v_{y_k}}(\hat{y}_k) = \mathcal{U}(v_{min_y}, v_{max_y})$ for y.

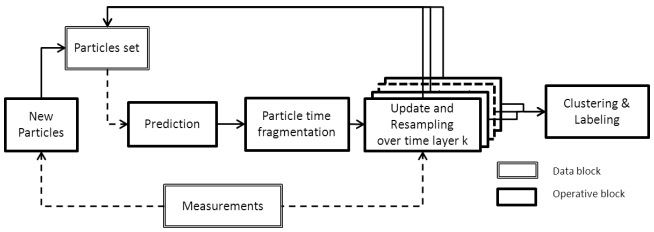


Fig. 1. Data (measurements and particles) and operations in a Particle-PHD (P-PHD) pipeline. The overlapping blocks represent the update and resampling steps performed over different time-strata.

The P-PHD pipeline is summarized in Fig. 1. An increase in targets, state dimension, clutter noise component leads to a growth in the number of particles that are necessary for a reliable tracking and consequently to an increase in the computational load. We aim to reduce this load by effectively spreading particles across processors, as described in the next section.

3. P-PHD PARALLELIZATION: THE P-BD FRAMEWORK

Let P be the number of available processors and $\{\omega_{k,p}^{(i)}, \boldsymbol{x}_{k,p}^{(i)}\}$ be the i-th particle, composed of its state and its weight allocated to processor $p \in \{1, 2, \ldots, P\}$. The complexity of the *prediction step* grows linearly with the number of particles that are independent of each other and therefore can be divided into different processors that can then compute the prediction independently. The computational complexity of *multimodal resampling* grows as $O(n^2)$ (n is the number of particles) and has a strict data dependence that make parallelization more challenging. In fact the data dependence shown in Eq. (2) implies additional communication among processors. Therefore we focus on the parallelization of the resampling step.

We propose a behavior-based parallelization combined with a distributed resampling (P-BD) with proportional allocation [9] (Fig.2). We aim to achieve a decomposition of the particles in subgroups to be analyzed separately in order to exploit the inequality $\sum_{j \in \mathcal{J}} n_j^2 < n^2$, where $n = \sum_{j \in \mathcal{J}} n_j$, $\mathcal{J}$ is the group of index j in which the total number of particles has been split and n_j is the number of particles in the j-th group. To this end, we cluster particles based on the target they belong to and obtain groups of particles that, if the targets are separated enough in the state space, can be considered independent, and therefore updated and resampled disjointly. We generate two types of fragmentations, namely an inter-processor fragmentation and an intra-processor fragmentation (fragmentation in different groups inside a processor).

The first step is the initialization, where the particles of each measurement are equally distributed among the processors to avoid unbalancing due to the elimination of particles associated with clutter measurements. The subsequent steps are done iteratively.

The *prediction* is performed with Eq. 1 over all particles independently. The *particle time fragmentation* block splits the particle set in S time-strata so they can be described by the tupla $\{\tilde{\omega}_k^{(i)} \tilde{\boldsymbol{x}}_k^{(i)}, s, p, l'\}$ where l' is the label that defines the macro-group of particles referring to different targets and treated jointly due to their mutual close proximity, and s and p are the strata and processor index. Each group is now defined as $\{\tilde{\omega}_k^{(i)} \tilde{\boldsymbol{x}}_k^{(i)}, s, p, l'\}_{i=1}^{L_{p,s,l'}}$ where $L_{p,s,l'}$ is the group cardinality.

The *update and resampling* steps operate as follows. Let $\Lambda_{s,p,l'}$=$\{\tilde{\omega}_k^{(i)} \tilde{\boldsymbol{x}}_k^{(i)}, s, p, l'\}_{i=1}^{L_{p,s,l'}}$ be the group of particles associ-

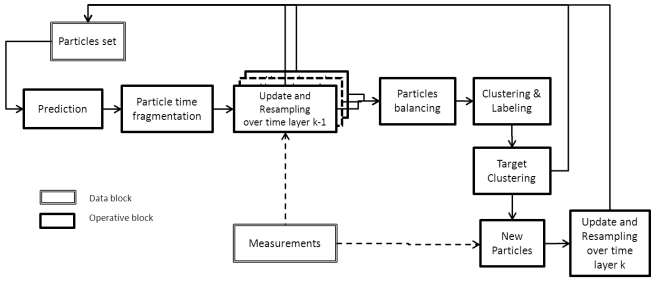


Fig. 2. P-PHD pipeline for P-BD: data blocks represent available measurement or particles. Data are sent to the operative blocks that performs the P-PHD. Prediction is executed in parallel. Overlapped blocks represent the update and resampling steps performed separately over different time-strata. For each strata the update and resampling are internally divided by particle group and processor using a distributed resampling with proportional allocation [9].

ated to p processors to which they belong, time layer s, and label l' while $\Gamma_{s,l'} = \{\Lambda_{s,p,l''} | l' = l''\}$ represents the group of particles with the same label l' assigned to different processors. $C_k(\boldsymbol{z})$ is computed with Eq. 3 over the particles of each group $\Gamma_{s,l'}$ and the complete measurement set. This is possible thanks to the negligible influence of (i) distant measurements over the updated particles and of (ii) distant particles with respect to the measurements interacting with the updated particles. MMR is also conducted separately over each $\Gamma_{s,l'}$ by means of the distributed resampling with proportional allocation [9] among the $\Lambda_{s,p,l'}$ sub-groups of $\Gamma_{s,l'}$. Unfortunately resampling may cause particle unbalancing. We therefore apply *particle routing* to rebalance the number of particles over each processor.

The clustering is done over all the particles that are then labelled with a group label l that substitutes the l' label. The total group of particles is now $\{\omega_k^{(i)} \boldsymbol{x}_k^{(i)}, s, p, l\}_{i=1}^{L_{p,s,l}}$. Each cluster represents a peak of the Gaussian Mixture Model introduced and a cluster list $\mathcal{C}$=$\{c_1, \ldots, c_L\}$ of estimated states c_j=$\{x, y, \dot{x}, \dot{y}\}$ where the index of each element corresponds to the cluster label l. We consider all the particles (of all available time-strata) because each particle needs a label to be associated to a target and to be correctly allocated to processors. Although we are using (for clustering) all the layers except the one at step k, multiple layers are useful to increase the number of particles for the real targets and reduce those for clusters associated to clutter (i.e. particles not propagated in other strata). In this manner clusters due to clutter can have a low weight and be removed by thresholding [7].

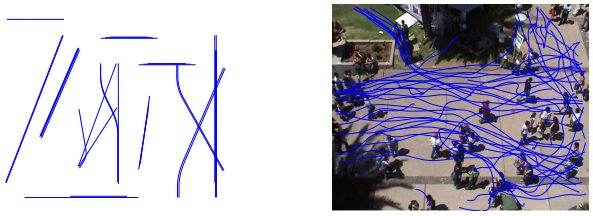
We need now to take into account possible *interacting* particles of targets in close proximity that should be analyzed contextually due to their dependence in the update and resampling step. The *target clustering* block aims to detect clusters in close proximity and to apply a second-level label l' to each particle so that in step $k+1$ particles with the same l' can be treated jointly.

Given the cluster list $\mathcal{C}$=$\{c_1, \ldots, c_L\}$ at time step k target clustering creates a new list $\tilde{\mathcal{C}}$=$\{\mathcal{C}_1, \ldots, \mathcal{C}_{L'}\}$ of subsets of $\mathcal{C}$ where L' is the number of subsets. We compute all possible distances between pairs and then group the elements of $\mathcal{C}$ in subsets. Note that a subset may be composed of a single target if the target is distant from others in the state space.

The grouping operation is based on the following constrains: $\forall c_i \in \mathcal{C}_j \nexists \mathcal{C}_p : c_i \in \mathcal{C}_p$; $\forall \{c_i, c_l\} \in \mathcal{C}_j \implies d\{c_i, c_l\} < d_{\mathbf{M}}$; and $\forall c_i \in \mathcal{C}_j, c_{i'} \in \mathcal{C}_{j'} \implies d\{c_i, c_{i'}\} > d_{\mathbf{M}}$. Then all particles whose label l corresponds to one of the indexes of the elements c_j of a target cluster $\mathcal{C}_\alpha$ are marked with the target cluster label $l' = \alpha$, leading to the particle $\{\omega_k^{(i)}, \boldsymbol{x}_k^{(i)}, p, l, l'\}$

The *allocation of new particles* is the last step of our approach. Now that each particle is labelled, we can label new particles according to the relation between measurement and actual multi-target state estimation. The elements of $\mathcal{C}$ are compared with the elements of $\boldsymbol{Z}_k$. For each pair $\{\boldsymbol{z}_i, c_j\}$ the Euclidean distance (over the position) $d_{i,j}$ is computed and $(d_{i,j}, i, j)$ is added to a list $\mathcal{A}$. A cut off distance $d_{\mathbf{MAX}}$ is applied and all the states whose distance is smaller than the threshold are deleted. The list is then sorted on the distance parameter. Starting from the low distance $d_{i',j'}$ a pair $\{i', j'\}$ is selected and moved to an empty list $\mathcal{B}$, and all the other vectors containing i' or j' are deleted from the list. The process is iterated to the end of the list. For each element of list $\mathcal{B}$, ρ particles are generated through the Eq. (5), centered on the measurement $\boldsymbol{z}_{i,k}$, labelled with j and allocated among processors. Additionally, also for not associated measurements, ρ particles are generated and labelled with available label indexes.

This pipeline enables intra-processor fragmentation while keeping the dependency on update and resampling between particles of the same target or group of targets.



(a) (b)

Fig. 3. Trajectories in datasets (a) D1 and (b) D2.

4. PERFORMANCE ASSESSMENT

We compare the proposed approach (P-BD) with two parallelization methods (P-NBC, P-NBD) for particle-based tracking [9] that differ in the way resampling is performed. The centralized resampling is used in P-NBC, whereas distributed resampling with proportional allocation is used in P-NBD. Both P-NBC and P-NBD perform parallelization without any contextual feedback from the tracking stage. The prediction step is executed independently for each particle. The update step requests the usual exchange of information between processors. In P-NBC all processors send all the particle weights to a single processor that performs the resampling and sends back the resampling factor for each particle. In P-NBD each processor computes the sum of the particles' weights and sends this value to one of the available processors that compares all the partial sums and computes the number of particles available for each processor in the resampling step. Next, each processor performs a separate resampling.

We use two datasets (D1 and D2) for the evaluation: D1 contains 32 point targets moving on a 7000×7000 meter plane following a constant velocity model and performing crossings or overtakings (Fig. 3a). The frame rate is 1Hz for 500 scans/frames, targets velocity and acceleration are between 0 and 10 m/s and 0 and 0.3 m/s^2, respectively. We uniformly add clutter measurements (the nearest integer to a value with mean $\mu_p = 4$ and variance $\sigma_p = 1$) in an area located between $0m$ and $500m$ meters in x position and $0m$ and $500m$ in y; D2 is the ground truth of a video sequence used in [10] (Fig. 3 b) of which we analyze the first 500 frames. The frame rate is

Table 1. Workload reduction (WR) between P-NBC and P-NBD (WR1) and between P-NBD and P-BD (WR2) in dataset D1 and D2 for varying number of processors (Pr.) and particles (Pa.).

		WR1 %		WR2 %			WR1 %		WR2 %	
Pr.	Pa.	D1	D2	D1	D2	Pa.	D1	D2	D1	D2
2	100	63	67	40	46	200	68	72	59	62
4	100	86	87	24	26	200	88	92	40	46
8	100	95	95	11	13	200	95	96	24	27

25Hz and analyzed on the image plane of 720 by 576 pixel. We add a random number (the nearest integer to a value normally distributed with mean $\mu_p = 4$ and variance $\sigma_p = 1$) of false positives in two areas: the first area is a rectangle located between 0 and 85 pixels in x position and between 320 and 576 in y and the second one between 320 and 450 pixels in x position and between 0 and 130 in y. The measurements used in D1 and D2 by the tracker are the sum of the targets' states plus white Gaussian noise with mean μ_n and variance σ_n. We corrupt the measurements to simulate the failure of a detector. A random number (the round of a normally distributed value with mean μ_c and variance σ_c) of false positive is added in the proximity of a random number (normally distributed with mean μ_{ct} and variance σ_{ct}) of measurements and where μ_c is obtained by the nearest integer of the percentage p_{ct} of targets in the scene in the considered instant. The clutter measurement is Gaussian distributed around the selected measurements with mean $\mu_e = 0$ and variance $\sigma_e = 0$. The parameters used are $\mu_n = 0$, $\sigma_n = 2$, $p_{ct} = 6$, $\sigma_{ct} = 1$, $\mu_c = 1$, $\sigma_c = 2$, $\mu_e = 0$ and $\sigma_e = 5$.

Performance evaluation is based on (i) the OSPA metric [11] to account for cardinality and accuracy errors, (ii) the number of particles moved across processors, and (iii) the total mean workload across all processors over time [12, 13]. The workload is evaluated over the resampling step, which is the different part in the methods under analysis and has a non-linear behavior with respect to the increase in the number of particles. The number of particles is associated to the workload with an empirical polynomial equation obtained by interpolating particles over resampling time. This analysis allows one to avoid potentially misleading results introduced by non-optimal code implementation of the parallel framework.

Figure 4 shows the results of the evaluation using D1 and D2. As for the *mean particle transfer* (Fig. 4 a,d) P-NBD outperforms P-NBC and P-BD due to a more straightforward resampling approach. The performance was evaluated on five trials. The *mean workload* (Fig.4 b,e) shows the most interesting results where in both D1 and D2 P-NBC has a higher workload value. In fact the resampling is done by a single processor and it does not change as a function of the number of processors. When the number of particles doubles, the workload triples due to the non-linear behavior discussed earlier. However, in both P-NBD and P-BD the workload decreases considerably. Table 1 (second and third columns) shows the workload reduction between P-NBC and P-NBD. The reduction is most significant when the number of particles is higher and in the case of configurations with more processors. The fragmentation among processors is useful for a large number of particles given the influence in the particles split in the non-linearity behavior of the resampling. The reduction rate is higher in D2 that has a larger number of targets. P-BD outperforms P-NBD, as shown in the last two columns of Table 1 due to the additional intra-processor fragmentation based on target independence. In this case the intra-processor fragmentation lead to an improved workload reduction when the number of particles grows. The reduction is larger in D2 when there are more targets and more particles, whereas it decreases as the number of processors

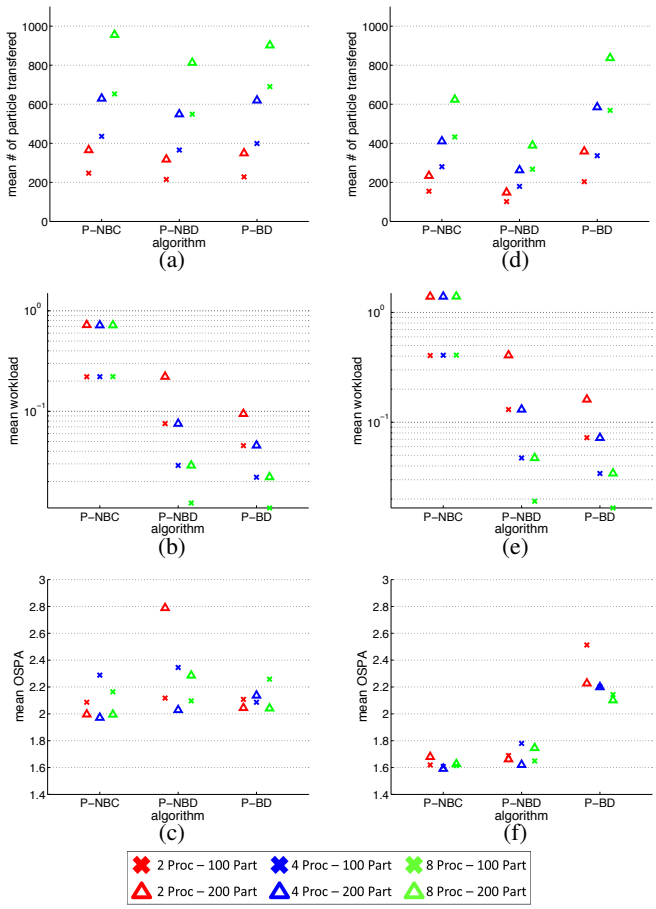


Fig. 4. Performance evaluation on dataset D1 (left column) and D2 (right column) for mean number of transferred particle (first row), mean workload (second row) and mean OSPA (third row).

increases as, with a higher number of processors, the usefulness of the intra-processor fragmentation becomes less significant.

In summary, P-NBC performs the resampling over a single processor showing the worst workload condition, whereas P-NBD uses the available processors showing a good workload reduction. The advantages of P-BD are directly proportional to the number of particles and inversely proportional to the number of processors. P-BD takes advantage of the tracking feedback to establish possible target independence and performs intra-resampling by fragmenting data, while reducing the computational complexity of resampling.

5. CONCLUSIONS

We proposed a P-PHD filter parallelization (P-BD) that is based on the exploitation of the distribution of the particles in the multi-target state space. The reduction of workload introduced by P-BD improves with the increase of the number of particles, however it is less useful when the number of processors grows because the influence of the intra-resampling becomes less significant. Future research directions include considering the allocation of particles among processor while reducing the amount of communication; the investigation of the limits over which the intra-fragmentation affects considerably accuracy; and the implementation of the proposed framework in a multi-processor platform [14].

6. REFERENCES

[1] Y. Bar-Shalom and T. E. Fortmann, *Tracking and Data Association*, Accademic Press, 1988.

[2] Y. Bar-Shalom, X. Rong Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*, John Wiley & Sons, 2001.

[3] E. Maggio and A. Cavallaro, *Video Tracking: Theory and Practice*, John Wiley and Sons, 2011.

[4] B.N. Vo, S.R. Singh, and A. Doucet., "Sequential monte carlo implementation of the phd filter for multi-target tracking.," in *In Proceedings of the International Conference on Information Fusion*, Cairns, Australia, 2003.

[5] R. Mahler, "Multi-target bayes filter via first-order multi-target moments.," *IEEE Transactions on Aereospace and Electronic Systems*, vol. 39, no. 4, pp. 1152–1178, April 2003.

[6] R. Mahler, "A theoretical foundation for the stein-winter probability hypothesis density (phd) multitarget tracking approach," in *Proceedings of Sensor and Data Fusion*, San Antonio, TX, 2000.

[7] E. Maggio, M. Taj, and A. Cavallaro, "Efficient multitarget visual tracking using random finite sets," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 18, no. 8, pp. 1016–1027, August 2008.

[8] Z. Shi, Y. Zheng, X. Bian, and Z. Yu, "Threshold based resampling for higt-speed particle phd filter," *Progress in Ectromagnetics Research*, vol. 136, pp. 369–383, 2013.

[9] M. Bolic, P.M. Djuric, and S. Hong, "Resampling algorithms and architectures for distributed particle filters," *IEEE Transactions on Signal Processing*, vol. 53, no. 7, pp. 2442–2450, 2005.

[10] J. Sochman and D. C. Hogg, "Who knows who - inverting the social force model for finding groups," in *Proc. of IEEE ICCVW*, Barcelona, Spain, 2011.

[11] B. Ristic, V. Ba-Ngu, and D. Clark, "Performance evaluation of multitarget tracking using the ospa metric," in *In Proceedings of Information Fusion*, Edinburgh, UK, July 2010.

[12] P. S. Pacheco, *An introduction to parallel programming*, Morgan Kaupmann, 2011.

[13] A. Grama, A. Gupta, G. Karypis, and V. Kumar, *Introduction to Parallel Computing*, Addison Wesley, 2003.

[14] P.G. Paulin, "OpenCL programming tools for the STHORM multi-processor platform: Application to computer vision," in *Proc. of International Forum on Embedded MPSoC and Multicore*, Otsu, Japan, 2013.