Adaptive auxiliary particle filter for track-before-detect with multiple targets

Luis Úbeda-Medina, *Student Member, IEEE*, Ángel F. García-Fernández, Jesús Grajal, *Member, IEEE*

***Abstract*—A novel particle filter for multiple target tracking with track-before-detect measurement models is proposed. Particle filters efficiently perform target tracking under nonlinear or non-Gaussian models. However, their application to multiple target tracking suffers from the curse of dimensionality. We introduce an efficient particle filter for multiple target tracking which deals with the curse of dimensionality better than previously developed methods. The proposed algorithm is tested and compared to other multiple target tracking particle filters.**

***Index Terms*—Particle filters, multiple target tracking, auxiliary particle filter, adaptive resampling, track-before-detect.**

I. Introduction

State estimation of dynamic systems is an extensively studied field, as it can be used to address a wide variety of problems in science and engineering [1]. In dynamic state estimation, the filtering problem consists of estimating the current state of the system given the observations, which are corrupted by measurement noise, obtained up to the current time.

The multiple target tracking (MTT) [2] problem comprises both the estimation of the number of targets in the surveillance area, and their individual states, which gather some kinematic information such as their positions and velocities. The general MTT problem can be posed in the Bayesian framework using random finite sets [3]. In the random finite sets framework the multitarget state is represented as a set whose elements are the single target state vectors. These state vectors can incorporate labels to allow for track formation.

In this paper, the interest lies in tracking multiple targets using track-before-detect measurement models, with special emphasis on the case that multiple targets get in close proximity [4], [5]. Due to the difficulty of the general MTT problem, we focus on the case of a fixed and known number of targets. In this case, representing the multitarget state as a vector is equivalent to a labeled set [6], [7]. Algorithms with this representation can be used in the general MTT setting to deal with the surviving targets after each time step, as done in [8], [9].

Luis Úbeda-Medina and Jesús Grajal are with the Departamento de Señales, Sistemas y Radiocomunicaciones, ETSI de Telecomunicación, Universidad Politécnica de Madrid, Ciudad Universitaria s/ n, 28040 Madrid, Spain. Email: luis.ubeda@upm.es, jesus.grajal@upm.es.

Ángel F. García-Fernández is with the Department of Electrical Engineering and Automation, Aalto University, 02150 Espoo, Finland. Email: angel.garciafernandez@aalto.fi.

Luis Úbeda-Medina was supported by the UPM fellowship program. This work was partially supported by project TEC2014-53815-R of the National Board of Scientific and Technological Research (CICYT), and by the Madrid Regional Government under project S2013/ICE-3000 (SPADERADAR-CM).

The Bayesian filtering approach to MTT aims to sequentially compute the posterior probability density function (PDF), which is the PDF of the current state given all measurements and contains all the information of interest about the current state of the targets. In general, the posterior cannot be described by a finite set of parameters, making its exact computation intractable except in some special cases [10].

Making use of importance sampling techniques, particle filters (PFs) can sequentially approximate the posterior PDF with any dynamic and measurement models. As the number of particles increases, the PF approximation to the posterior becomes more accurate [11], [12], [13]. Nevertheless, efficiently sampling high dimensional state spaces, as is usually the case in MTT, is generally quite difficult.

It was shown in [14] that, in order to reduce the curse of dimensionality, it is useful to assume that the current target states are independent. This posterior independence assumption can improve the PF approximation to the prior at the next time step, especially for a low enough number of particles or a high number of targets. Examples of PFs with this assumption are the independent partition PF (IP) [15] or the parallel partition PF (PP) [8]. In addition to making this assumption, one should carefully select the importance density, which determines how target states are sampled. As an example, IP does not consider neighboring targets in the sampling procedure of each individual target, therefore resulting in a severe performance loss when targets move in close proximity. In order to overcome this shortcoming, PP incorporates an estimation of the state of the neighboring targets in the sampling procedure.

Another option to tackle the curse of dimensionality is to approximate the marginal PDF for each target, as done in [16] or in the multiple PF (MPF) [17], [18], [19]. However, it is sometimes useful to obtain the updated joint PDF of the target states [20].

None of the above PFs obtain samples from the optimal importance density, which minimizes the variance of the particle weights [21]. The independent joint optimal importance density (IJOID) [14] uses the optimal importance density to sample from the state space once the posterior independence assumption has been made to tackle the curse of dimensionality. However, IJOID cannot be applied to any track-before-detect measurement model [22], so it does not result in a generally applicable algorithm, which is the aim of this paper.

The auxiliary particle filter (APF) [23] aims to mimic the way samples are produced by the optimal importance density by taking into account the actual measurements by means of an auxiliary variable. However, the performance of the APF

rapidly decays with an increase in the number of targets, suffering from the curse of dimensionality as it jointly samples the whole state space.

In this paper, we first propose two particle filters that generalize the APF for MTT using the posterior independence assumption and the state partition strategy to overcome the curse of dimensionality. The main difference between them is that the first filter, the target-resampling auxiliary PP (TRAPP), includes a target-resampling stage and the second filter, the auxiliary PP (APP), does not. A preliminary version of these results was presented in [24]. We then analyze why the filter with the best performance changes between these two depending on the scenario, demonstrating that the use of the target-resampling procedure is especially useful when dealing with high-dimensional state spaces.

It is a novelty of this paper to consider target-resampling within an update step to adequately handle large dimensional state estimation problems. However, target-resampling reduces sample diversity, meaning that it does not always improve performance. Thus, the last contribution of this paper is the development of a new algorithm that is able to adaptively combine both techniques: the adaptive TRAPP (ATRAPP). Contrary to the IJOID method, the presented algorithm does not need to be constrained to its use with any particular track-before-detect measurement model, thus being a general method for nonlinear filtering. ATRAPP gathers the best features of APP and TRAPP and, consequently, is able to outperform both of them. ATRAPP monitors the effective sampling size [22] for each target to adaptively decide if target-resampling is needed. This makes ATRAPP a robust algorithm with a reliable performance regardless of the number of targets being tracked. In the final part of the paper we provide extensive numerical results to evaluate the performance of ATRAPP in comparison with state-of-the-art filters for track-before-detect.

The rest of the paper is organized as follows: In Section II, the problem of tracking a fixed and known number of targets is formulated. In Section III, we thoroughly explain APP and TRAPP. In Section IV, we analyze the suitability of target-resampling for high target number. In Section V, the ATRAPP PF is introduced. In Section VI, the performance of ATRAPP is compared with other PFs via simulations. Finally, conclusions are drawn in Section VII.

II. Problem Statement

In this section, we describe the MTT problem we consider. The following two assumptions are made:

- Assumption A: There is a fixed and known number of targets in the surveillance area.
- Assumption B: Targets move independently.

Under Assumption A, the state of the targets at time k can be described by the multitarget state vector

$$\mathbf{X}^k = \left[(\mathbf{x}_1^k)^T, (\mathbf{x}_2^k)^T, ..., (\mathbf{x}_t^k)^T\right]^T \in \mathbb{R}^{n \cdot t} \tag{1}$$

where t stands for the number of targets, the superscript T stands for the vector transpose and the state of target j at time k is described by the n-dimensional target state vector $\mathbf{x}_j^k$, which comprises kinematic features of the target such as its position and velocity. Under Assumptions A and B the dynamic system is described by the state-transition equations of the targets and the measurement equation

$$\mathbf{x}_j{}^{k+1} = \mathbf{f}_j(\mathbf{x}_j^k, \mathbf{w}_j^k) \tag{2}$$

$$\mathbf{z}^{k+1} = \mathbf{h}(\mathbf{X}^{k+1}, \mathbf{v}^{k+1}) \tag{3}$$

where $\mathbf{f}_j(\cdot,\cdot)$ and $\mathbf{h}(\cdot,\cdot)$ might be nonlinear functions, $\mathbf{w}_j^k$ is the process noise vector for the j-th target at time k, $\mathbf{z}^{k+1}$ denotes the observation at time $k+1$, and $\mathbf{v}^{k+1}$ is the measurement noise vector at time $k+1$. Noise vectors at each time step are assumed to be independent. Note that Equation (2) implicitly assumes that the system evolves according to a Markov process.

In the Bayesian setting, MTT is based on the recursive approximation of the joint multitarget posterior PDF, $p(\mathbf{X}^k|\mathbf{z}^{1:k})$, where $\mathbf{z}^{1:k}$ represents a sequence of measurements $\mathbf{z}^1, \mathbf{z}^2, ..., \mathbf{z}^k$ taken from time 1 to time k. Following the usual steps of Bayesian filtering, the posterior PDF of the state at time $k+1$ based on the measurements up to time $k+1$, can be obtained by recursively applying two steps: prediction and update [25].

In the prediction step, the prior PDF at time $k+1$, which denotes the PDF of the current state given the measurements up to time k, is computed via the Chapman-Kolmogorov equation

$$p(\mathbf{X}^{k+1}|\mathbf{z}^{1:k}) = \int p(\mathbf{X}^k|\mathbf{z}^{1:k}) p(\mathbf{X}^{k+1}|\mathbf{X}^k) d\mathbf{X}^k \tag{4}$$

$$= \int p(\mathbf{X}^k|\mathbf{z}^{1:k}) \prod_{j=1}^{t} p(\mathbf{x}_j{}^{k+1}|\mathbf{x}_j^k) d\mathbf{X}^k \tag{5}$$

where the transition PDFs, $p(\mathbf{x}_j^{k+1}|\mathbf{x}_j^k)$, are obtained from equation (2). Once the measurement at time $k+1$ is available, the update step makes use of Bayes' rule to provide the posterior at time $k+1$

$$p(\mathbf{X}^{k+1}|\mathbf{z}^{1:k+1}) \propto p(\mathbf{z}^{k+1}|\mathbf{X}^{k+1}) p(\mathbf{X}^{k+1}|\mathbf{z}^{1:k}) \tag{6}$$

where $\propto$ indicates proportionality and the PDF $p(\mathbf{z}^{k+1}|\mathbf{X}^{k+1})$ can be obtained from the measurement equation in (3). However, when the dynamic or measurements are nonlinear or non-Gaussian, the posterior PDF (6) does not generally admit a closed-form expression, so approximations are necessary. In the next section, some PF approximations to this recursion are proposed.

III. Auxiliary state partition PFs

The dimension of the state in the MTT problem grows linearly with the number of targets. This poses a significant challenge for PFs as they typically require a prohibitively large number of particles to deal with a state of sufficiently high dimension. This effect is commonly referred to as the curse of dimensionality [26], [27]. As mentioned in the introduction, a common strategy to alleviate this curse of dimensionality is to use a posterior independence assumption [14], in which targets are assumed to be independent in the posterior at the

previous time step. The methods presented in this paper also make use of this approach.

In this section, the auxiliary PP (APP) PF and the target-resampling APP (TRAPP) PF are presented. We also highlight their advantages and limitations in order to illustrate the benefits of the adaptive procedure presented in Section V.

Particle filters approximate the posterior PDF, $p(\mathbf{X}^{k+1}|\mathbf{z}^{1:k+1})$, by sequentially propagating a set of N particles with their associated weights, $\{(\mathbf{X}_1^{k+1}, \omega_1^{k+1}), ..., (\mathbf{X}_N^{k+1}, \omega_N^{k+1})\}$, based on the proposal of an importance sampling density function, $q(\mathbf{X}^{k+1}|\mathbf{z}^{1:k+1})$. Therefore, the update of the weights is performed according to the principle of importance sampling

$$\omega_i^{k+1} \propto \frac{p(\mathbf{X}_i^{k+1}|\mathbf{z}^{1:k+1})}{q({\mathbf{X}_i}^{k+1}|\mathbf{z}^{1:k+1})} \tag{7}$$

$$\propto \frac{p(\mathbf{z}^{k+1}|\mathbf{X}_i^{k+1})p({\mathbf{X}_i}^{k+1}|\mathbf{z}^{1:k})}{q({\mathbf{X}_i}^{k+1}|\mathbf{z}^{1:k+1})} \tag{8}$$

Particle $\mathbf{X}_i^k$ is

$$\mathbf{X}_i^k = \left[(\mathbf{x}_{1,i}^k)^T, ..., (\mathbf{x}_{t,i}^k)^T\right]^T \in \mathbb{R}^{n\cdot t}$$

where $\mathbf{x}_{j,i}^k$ is the part of the particle that corresponds to target j, which is referred to as a subparticle. Each particle is therefore formed by t subparticles. To describe the proposed methods, we find the following nomenclature useful. In the prediction and update steps from time k to time $k+1$, particles and subparticles at time k are referred to as *parent* particles and subparticles, while particles and subparticles at time $k+1$ are referred to as *child* particles and subparticles.

The filters we propose in this paper are based on the posterior independence assumption so we proceed to write the posterior for this case. Given a particle filter approximation to the posterior at time k, and considering posterior independence assumption, the posterior PDF at time $k+1$ becomes

$$p(\mathbf{X}^{k+1}|\mathbf{z}^{1:k+1}) \propto p(\mathbf{z}^{k+1}|\mathbf{X}^{k+1}) \prod_{j=1}^{t} \sum_{i=1}^{N} \omega_i^k p(\mathbf{x}_j^{k+1}|\mathbf{x}_{j,i}^k) \tag{9}$$

In order to allow for the developed filters to account for neighboring targets in the sampling step, we denote the averaged predicted target state for target j as

$$\hat{\mathbf{x}}_j^{k+1} = \sum_{i=1}^{N} \omega_i^k \cdot \mathbf{x}_{j,i}^{k+1|k} \tag{10}$$

where $\mathbf{x}_{j,i}^{k+1|k}$ is the prediction of the j-th target state at time $k+1$ of the i-th subparticle at time k [28]. For each target, we build the vector $\hat{\mathbf{X}}_{-\{j\}}^{k+1}$ stacking the predicted state at time $k+1$ of every target except for target j as

$$\hat{\mathbf{X}}_{-\{j\}}^{k+1} = \left[(\hat{\mathbf{x}}_1^{k+1})^T, ..., (\hat{\mathbf{x}}_{j-1}^{k+1})^T, (\hat{\mathbf{x}}_{j+1}^{k+1})^T, ..., (\hat{\mathbf{x}}_t^{k+1})^T\right]^T$$

We now define the predicted target likelihood, $b_j(\mathbf{x}_j^{k+1})$, as

$$b_j(\mathbf{x}_j^{k+1}) \propto p(\mathbf{z}^{k+1}|\hat{\mathbf{X}}_{-\{j\}}^{k+1}, \mathbf{x}_j^{k+1}), \tag{11}$$

which takes into account the rest of the targets by their predicted states.

A. APP PF

APP obtains samples from the posterior by making use of an auxiliary variable [23] for each target, therefore sampling in a higher dimension from

$$p(\mathbf{X}^{k+1}, \mathbf{a}|\mathbf{z}^{1:k+1}) \propto p(\mathbf{z}^{k+1}|\mathbf{X}^{k+1}) \prod_{j=1}^{t} \omega_{a_j}^k p(\mathbf{x}_j^{k+1}|\mathbf{x}_{j,a_j}^k) \tag{12}$$

where $\mathbf{a} = [a_1, a_2, ..., a_t]^T \in \{1, 2, ..., N\}^t$, and a_j is an index in the mixture of the j-th target in (9). The selection of the auxiliary variable is sound because $p(\mathbf{X}^{k+1}, \mathbf{a}|\mathbf{z}^{1:k+1}) \geq 0$ and by integrating (12) over $\mathbf{a}$, one gets the posterior.

The use of an auxiliary variable decreases the computational burden and enables subparticle crossover. Subparticle crossover is a desirable property in such filters, as it allows the formation of a particle at time $k+1$ from the propagation of subparticles that belonged to different particles at time k. Using an auxiliary variable, APP performs the crossover between parent subparticles. This is, particles at $k+1$ are formed based on a selection of the parent subparticles that are prone to obtain child subparticles with higher likelihoods.

As previously mentioned, samples are produced from an importance density function. In the case of APP, the importance density function samples targets independently and is selected as

$$q^A(\mathbf{X}^{k+1}, \mathbf{a}|\mathbf{z}^{1:k+1}) = \prod_{j=1}^{t} q_j^A(\mathbf{x}_j^{k+1}, a_j|\mathbf{z}^{1:k+1}) \tag{13}$$

$$q_j^A(\mathbf{x}_j^{k+1}, a_j|\mathbf{z}^{1:k+1}) \propto b_j(\boldsymbol{\mu}_{j,a_j}^{k+1})\omega_{a_j}^k p(\mathbf{x}_j^{k+1}|\mathbf{x}_{j,a_j}^k)$$

where $b_j(\boldsymbol{\mu}_{j,a_j}^{k+1})\omega_{a_j}^k$ constitute the first-stage weights of the APP filter, with $\boldsymbol{\mu}_{j,a_j}^{k+1}$ being some characterization of $\mathbf{x}_j^{k+1}$ given $\mathbf{x}_{j,a_j}^k$, such as the predicted mean, $\boldsymbol{\mu}_{j,a_j}^{k+1} = \mathrm{E}[\mathbf{x}_j^{k+1}|\mathbf{x}_{j,a_j}^k]$, or a sample, $\boldsymbol{\mu}_{j,a_j}^{k+1} \sim p(\mathbf{x}_j^{k+1}|\mathbf{x}_{j,a_j}^k)$.

Substituting Equation (13) into (7), the updated weights become

$$\omega_i^{k+1} \propto \frac{p(\mathbf{z}^{k+1}|\mathbf{X}_i^{k+1}) \prod_{j=1}^{t} \omega_{a_j^i}^k p(\mathbf{x}_{j,i}^{k+1}|\mathbf{x}_{j,a_j^i}^k)}{\prod_{j=1}^{t} b_j(\boldsymbol{\mu}_{j,a_j^i}^{k+1})\omega_{a_j^i}^k p(\mathbf{x}_{j,i}^{k+1}|\mathbf{x}_{j,a_j^i}^k)}$$

$$\propto \frac{p(\mathbf{z}^{k+1}|\mathbf{X}_i^{k+1})}{\prod_{j=1}^{t} b_j(\boldsymbol{\mu}_{j,a_j^i}^{k+1})} \tag{14}$$

Note that for one target, $t = 1$, APP is in fact the conventional auxiliary particle filter [23]. Under the conditions stated in [23], i.e. informative measurements, the use of this method to obtain the samples from the importance sampling function allows for the reduction of the variance of the first-stage weights. Thus, APP obtains a better approximation than PP [8] of the importance sampling function in (16) for a fixed number of samples. A pseudocode of the APP method is given in Algorithm 1.

It should be noted that as a consequence of the definition of $b_j(\mathbf{x}_j^{k+1})$, the APP method can always be applied to

approximate the prediction (4) and update (6) equations, so that it can be satisfactorily applied to any other Bayesian filtering problem under Assumptions A and B.

B. TRAPP

In this section we present the TRAPP PF. The main difference with APP is that TRAPP uses a resampling step for each target, which is useful in some situations, as will be analyzed in Section IV.

As happened for APP, TRAPP also uses the posterior independence assumption and individually samples each target state. Subparticle crossover is again desired, so TRAPP also makes use of an auxiliary variable for each target, therefore sampling in a higher dimension from (12). The importance sampling function proposed for the TRAPP method takes the form

$$q^T(\mathbf{X}^{k+1}, \mathbf{a}|\mathbf{z}^{1:k+1}) = \prod_{j=1}^{t} q_j^T(\mathbf{x}_j^{k+1}, a_j|\mathbf{z}^{1:k+1}) \quad (15)$$

$$q_j^T(\mathbf{x}_j^{k+1}, a_j|\mathbf{z}^{1:k+1}) \propto b_j(\mathbf{x}_j^{k+1})\omega_{a_j}^k p(\mathbf{x}_j^{k+1}|\mathbf{x}_{j,a_j}^k) \quad (16)$$

where we recall that $b_j(\mathbf{x}_j^{k+1})$ is given by (11) to account for nearby targets. Substituting (16) in (7), the updated weight is given by

$$\begin{aligned}\omega_i^{k+1} &\propto \frac{p(\mathbf{z}^{k+1}|\mathbf{X}_i^{k+1})\prod_{j=1}^{t}\omega_{a_j^i}^k p(\mathbf{x}_{j,i}^{k+1}|\mathbf{x}_{j,a_j^i}^k)}{\prod_{j=1}^{t} b_j(\mathbf{x}_{j,i}^{k+1})\omega_{a_j^i}^k p(\mathbf{x}_{j,i}^{k+1}|\mathbf{x}_{j,a_j^i}^k)} \\ &= \frac{p(\mathbf{z}^{k+1}|\mathbf{X}_i^{k+1})}{\prod_{j=1}^{t} b_j(\mathbf{x}_{j,i}^{k+1})} \end{aligned} \quad (17)$$

However, as stated in [8] for the IP and PP filters, it is not generally feasible to sample directly from (16). TRAPP therefore makes use of the sampling/importance resampling method in [29] to obtain samples from (16), thus sampling from a first-pass approximation of (16)

$$h_j(\mathbf{x}_j^{k+1}, a_j|\mathbf{z}^{1:k+1}) \propto b_j(\boldsymbol{\mu}_{j,a_j}^{k+1})\omega_{a_j}^k p(\mathbf{x}_j^{k+1}|\mathbf{x}_{j,a_j}^k) \quad (18)$$

where, as before, $\boldsymbol{\mu}_{j,a_j}^{k+1}$ is some characterization of $\mathbf{x}_j^{k+1}$given $\mathbf{x}_{j,a_j}^k$ such as the predicted mean or a sample. A set of M samples $(\mathbf{x}_j^*, a_j^*)$ are drawn from this first-pass approximation and the following quotient is computed

$$r(\mathbf{x}_{j,p}^*, \mathbf{a}_{j,p}^*) \propto \frac{q_j^T(\mathbf{x}_{j,p}^*, a_{j,p}^*|\mathbf{z}^{1:k+1})}{h_j(\mathbf{x}_{j,p}^*, a_{j,p}^*|\mathbf{z}^{1:k+1})} = \frac{b_j(\mathbf{x}_{j,p}^*)}{b_j(\boldsymbol{\mu}_{j,p}^*)} \quad (19)$$

in order to draw N samples of $q_j^T(\mathbf{x}_j^{k+1}, \mathrm{a}_j|\mathbf{z}^{1:k+1})$ from the distribution defined by $(\mathbf{x}_{j,p}^*, a_{j,p}^*)$, for $p = 1, 2..., M$ (it is common to set $M = N$) with probability $r(\mathbf{x}_{j,p}^*, a_{j,p}^*)$. The drawn samples for each target are then introduced in (17) to compute the particle weights.

Hence, the TRAPP PF selects for every target the parent subparticles at time k that are prone to generate child subparticles with a higher likelihood given the measurement at time $k+1$ based on the auxiliary filtering in Equation (18). This causes the crossover of *parent* subparticles. Then, TRAPP performs crossover of the *child* subparticles. That is, TRAPP selects for every target those *child* subparticles $\mathbf{x}_{j,p}^*$ that show a higher likelihood with the measurement at time $k+1$, performing a per target-resampling using Equation (19). Finally the particle weights are computed using Equation (17).

Note the clear resemblance of the described procedure for drawing samples from (16) to the APF. The TRAPP PF is a generalization of the APF for multiple target tracking. Note that for $t = 1$ TRAPP is in fact the same filter presented in [23] with a resampling stage. A pseudocode of the TRAPP method is pointed out in Algorithm 2.

With the aim of clarifying the details and differences of TRAPP, APP, PP [8] and IP [15], [30] and how they perform subparticle crossover, Table I indicates the different properties of these algorithms.

Table I: Properties of IP, PP, APP and TRAPP algorithms.

MTT filter	Generalization of APF to MTT	target-resampling	accounts for nearby targets	avoids particle resampling
IP	×	✓	×	×
PP	×	✓	✓	×
APP	✓	×	✓	✓
TRAPP	✓	✓	✓	✓

IV. On the use of resampling

In this section, we motivate the possible benefits of using target-resampling, as in TRAPP, compared to not using it, as in APP. In general, resampling is used in particle filters to avoid particle degeneracy that occurs over time [22]. Without resampling, the variance of the particle weights increases until there is only one particle with a non-negligible weight. Clearly, this is not an accurate representation of the posterior PDF. The underlying cause is that the particle filter is in fact approximating the posterior PDF over all trajectories, which implies that the dimension of the state grows at each time step [31]. This degeneracy of particle weights is therefore associated to an increase in the dimension of the state under consideration and its effects can be mitigated by resampling.

In MTT, the dimensionality of the state space depends on the number of targets being tracked. In this sense, particle degeneracy arises in the sampling of high dimensional space-time states, which can be dealt with target-resampling. As such, in this paper, we use the concept of target-resampling to overcome degeneracy in MTT filters, either this being a consequence of the number of targets being tracked or of the number of steps of their trajectories.

It also should be taken into account that resampling produces a side-effect, commonly referred to as sample-impoverishment [25], meaning that there will be a loss of diversity in the resultant resampled approximation to the PDF, as it will contain repeated samples. We find it useful to illustrate these insights with the following example.

Algorithm 1 APP procedure

procedure $(\mathbf{z}^{k+1}, \hat{\mathbf{X}}^{k+1}_{-\{j\}}, \{(\mathbf{X}^k_1, \omega^k_1), (\mathbf{X}^k_2, \omega^k_2), ..., (\mathbf{X}^k_N, \omega^k_N)\})$

for $j = 1, ..., t$ **do** ▷ **Auxiliary filtering**
for $i = 1, ..., N$ **do**
- Draw $\boldsymbol{\mu}^{k+1}_{j,i}$ using:
$p(\mathbf{x}^{k+1}_j|\mathbf{x}^k_{j,i})$ or $\mathrm{E}[\mathbf{x}^{k+1}_j|\mathbf{x}^k_{j,i}]$
- Compute $b_j(\boldsymbol{\mu}^{k+1}_{j,i}) = p(\mathbf{z}^{k+1}|\hat{\mathbf{X}}^{k+1}_{-\{j\}}, \boldsymbol{\mu}^{k+1}_{j,i})$
- Compute $\lambda_{j,i} = b_j(\boldsymbol{\mu}^{k+1}_{j,i})\omega^k_i$
end for
for $i = 1, ..., N$ **do**
- Normalize $\lambda_{j,i} = \frac{\lambda_{j,i}}{\sum_{i=1}^N \lambda_{j,i}}$
end for
for $i = 1, ..., N$ **do**
- Sample an index a^i_j from the distribution defined by $(\lambda_{j,1}, \lambda_{j,2}, ..., \lambda_{j,N})$
- Draw a sample $\mathbf{x}^{k+1}_{j,i}$ using $p(\mathbf{x}^{k+1}_j|\mathbf{x}^k_{j,a^i_j})$
end for
end for

for $i = 1, ..., N$ **do** ▷ **Weight update**
- Compute $\omega^{k+1}_i = \frac{p(\mathbf{z}^{k+1}|\mathbf{X}^{k+1}_i)}{\prod_{j=1}^t b_j(\boldsymbol{\mu}^{k+1}_{j,a^i_j})}$
end for
for $i = 1, ..., N$ **do**
- Normalize $\omega^{k+1}_i = \frac{\omega^{k+1}_i}{\sum_{i=1}^N \omega^{k+1}_i}$
end for

return $\{(\mathbf{X}^{k+1}_1, \omega^{k+1}_1), (\mathbf{X}^{k+1}_2, \omega^{k+1}_2), ..., (\mathbf{X}^{k+1}_N, \omega^{k+1}_N)\}$

end procedure

Algorithm 2 TRAPP procedure

procedure $(\mathbf{z}^{k+1}, \hat{\mathbf{X}}^{k+1}_{-\{j\}}, \{(\mathbf{X}^k_1, \omega^k_1), (\mathbf{X}^k_2, \omega^k_2), ..., (\mathbf{X}^k_N, \omega^k_N)\})$

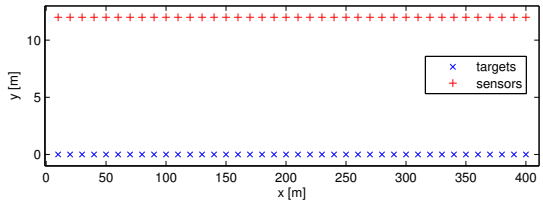
for $j = 1, ..., t$ **do** ▷ **Auxiliary filtering**
for $i = 1, ..., N$ **do**
- Draw $\boldsymbol{\mu}^{k+1}_{j,i}$ using:
$p(\mathbf{x}^{k+1}_j|\mathbf{x}^k_{j,i})$ or $\mathrm{E}[\mathbf{x}^{k+1}_j|\mathbf{x}^k_{j,i}]$
- Compute $b_j(\boldsymbol{\mu}^{k+1}_{j,i}) = p(\mathbf{z}^{k+1}|\hat{\mathbf{X}}^{k+1}_{-\{j\}}, \boldsymbol{\mu}^{k+1}_{j,i})$
- Compute $\lambda_{j,i} = b_j(\boldsymbol{\mu}^{k+1}_{j,i})\omega^k_i$
end for
for $i = 1, ..., N$ **do**
- Normalize $\lambda_{j,i} = \frac{\lambda_{j,i}}{\sum_{i=1}^N \lambda_{j,i}}$
end for
for $i = 1, ..., N$ **do**
- Sample an index a^i_j from the distribution defined by $(\lambda_{j,1}, \lambda_{j,2}, ..., \lambda_{j,N})$
- Draw a sample $\mathbf{x}^*_{j,a^i_j}$ using $p(\mathbf{x}^{k+1}_j|\mathbf{x}^k_{j,a^i_j})$
- Compute $b_j(\mathbf{x}^*_{j,a^i_j}) = p(\mathbf{z}^{k+1}|\hat{\mathbf{X}}^{k+1}_{-\{j\}}, \mathbf{x}^*_{j,a^i_j})$
- Compute $r_j(\mathbf{x}^*_{j,a^i_j}) = \frac{b_j(\mathbf{x}^*_{j,a^i_j})}{b_j(\boldsymbol{\mu}^{k+1}_{j,a^i_j})}$
end for
for $i = 1, ..., N$ **do**
- Normalize $r_j(\mathbf{x}^*_{j,a^i_j}) = \frac{r_j(\mathbf{x}^*_{j,a^i_j})}{\sum_{i=1}^N r_j(\mathbf{x}^*_{j,a^i_j})}$
end for
for $i = 1, ..., N$ **do** ▷ **target-resampling**
- Sample an index p from the distribution defined by $(r_j(\mathbf{x}^*_{j,a^1_j}), r_j(\mathbf{x}^*_{j,a^2_j}), ..., r_j(\mathbf{x}^*_{j,a^N_j}))$
- Set $\mathbf{x}^{k+1}_{j,i} = \mathbf{x}^*_{j,a^p_j}$
end for
end for
for $i = 1, ..., N$ **do** ▷ **Weight Update**
- Compute $\omega^{k+1}_i = \frac{p(\mathbf{z}^{k+1}|\mathbf{X}^{k+1}_i)}{\prod_{j=1}^t b_j(\mathbf{x}^{k+1}_{j,i})}$
end for
for $i = 1, ..., N$ **do**
- Normalize $\omega^{k+1}_i = \frac{\omega^{k+1}_i}{\sum_{i=1}^N \omega^{k+1}_i}$
end for

return $\{(\mathbf{X}^{k+1}_1, \omega^{k+1}_1), (\mathbf{X}^{k+1}_2, \omega^{k+1}_2), ..., (\mathbf{X}^{k+1}_N, \omega^{k+1}_N)\}$

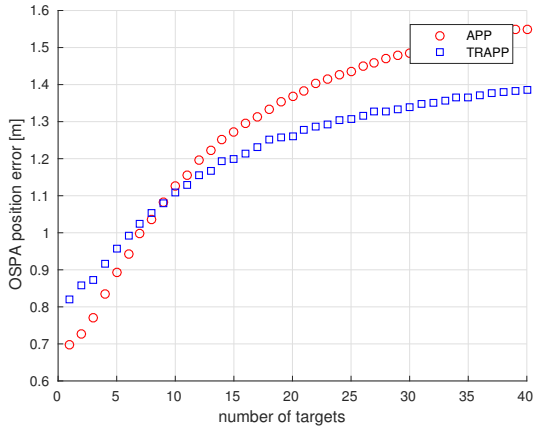
end procedure

Example 1. A different number of targets ranging from $t = 1$ to $t = 40$ are considered. In each simulation, targets are arranged in a straight line so that for $j = 1, ..., t$, the j-th target position is given by $\mathbf{p}_j = [j \cdot d_t, 0]^T$, with $d_t = 10$m. A total of $N_s = 40$ sensors are displayed in the scenario at positions $\mathbf{s}_i = [i \cdot d_t, 12]^T$, where $i = 1, ..., 40$ and the nonlinear measurement model described in Section VI, using a measurement noise variance of $\sigma^2_s = 2$. To simplify the experiment, we only consider an update step; the a priori position density function for each target, $\mathcal{N}(\mathbf{p}_j; \bar{\mathbf{p}}_j \boldsymbol{\Sigma})$, is a Gaussian PDF with mean $\bar{\mathbf{p}}_j$ corresponding to the real target position, and covariance matrix $\Sigma = \sigma^2 I_2$, where $\mathbf{I}_n$ is the $n \times n$ identity matrix and $\sigma = 1m$. In Figure 1(b), we show the optimal sub-pattern assignment [32] (OSPA) position error (with parameters $p = 2$, $c = 10$m) for a different number of targets averaged in a Monte Carlo simulation with $n = 10000$ runs. It can be observed that if the dimension of the state space is high enough (9 targets in this case), the target-resampling stage makes it possible for TRAPP to achieve a lower error than APP. On the other hand, for a low number of targets APP outperforms TRAPP. This is to be expected as the dimensionality of the state is lower so target-resampling is counterproductive.

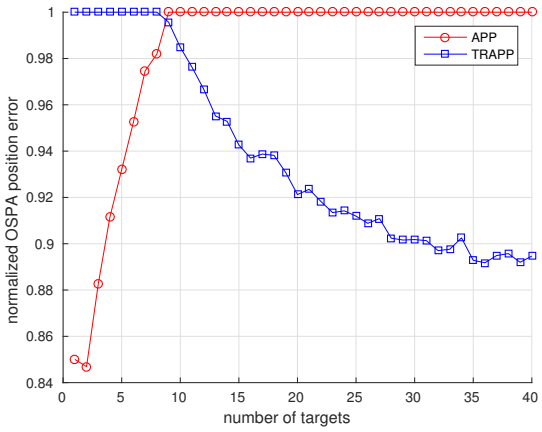
In order to clarify this, we also show in Figure 1(c), the normalized OSPA position errors with respect to the worst performing filter. In this simple example, APP obtains an error up to a 14% lower than TRAPP for a low number of targets, while TRAPP obtains an error of almost an 11% lower than APP does when considering a high enough number of targets. Note that, for the sake of simplicity, this example uses only a one-step update. However, if we use these algorithms in a normal filtering set-up with a recursive approximation of



(a)



(b)



(c)

Figure 1: (a) Proposed scenario for Example 1 with up to $t = 40$ targets. (b) OSPA position error as a function of the number of targets for the APP and TRAPP algorithms for one update step. (c) Normalized OSPA position error with respect to the worst performing filter in each case. It is shown how the target-resampling stage of TRAPP is advisable for a high enough number of targets, while it is not for a low number of targets.

the filtering PDF, errors propagate over time. This means that even small differences in the errors in one update can result in bigger differences as time goes on, as shown in the simulation examples of Section VI.

In this section we have analyzed target-resampling as a possibly good feature for a multitarget PF. However, performing an additional resampling for the whole particles, which we call particle-resampling (see Table I), as done in IP and PP, is generally detrimental as we would be resampling particles whose subparticles have already been resampled. This double resampling generally implies a loss of particle diversity that can be avoided with the algorithms presented in this paper.

V. ATRAPP

We have seen in the previous section that target-resampling becomes an advantage for a high number of targets. However, in general, it is difficult to know when it is beneficial to perform target-resampling or not, as it depends on all the parameters of the problem at hand. In order to tackle this issue, in this section, we develop the adaptive TRAPP (ATRAPP) particle filter. ATRAPP aims to combine the best features of both APP and TRAPP in a single algorithm, where the decision of performing target-resampling is independently taken for each target.

A common approach to decide whether to resample in PFs is the monitoring of the effective sample size [33], [21], which measures the degeneracy of the particle weights. Then, we perform target-resampling if the effective sample size falls below a given threshold. We proceed to extend this technique for target-resampling. This will enable us to combine the strengths of APP and TRAPP into ATRAPP.

As previously stated, the target-resampling stage in TRAPP is achieved by taking samples from the distribution defined by $(\mathbf{x}^*_{j,p}, a^*_{j,p})$, for $p = 1, 2..., M$ with the normalized weights $r(\mathbf{x}^*_{j,p}, a^*_{j,p})$ in (19). ATRAPP measures subparticle degeneracy using this distribution (with $M = N$), computing the effective target sample size, N^j_{eff}, to decide if target-resampling is really advisable for target j. One cannot generally exactly evaluate N^j_{eff}, however a commonly accepted estimate of the effective sample size is given by [22]

$$\hat{N}^j_{eff} = \frac{1}{\sum_{p=1}^{N} r^2(\mathbf{x}^*_{j,p}, a^*_{j,p})} \tag{20}$$

It should be noted that $\hat{N}^j_{eff}$ can take values in the interval $[1, N]$. If this estimate of the effective target sample size drops below a threshold, target-resampling is performed among the child subparticles of target j as in TRAPP, avoiding the subparticle degeneracy that would lead the whole filter to an undesired global particle degeneracy. On the other hand, if $\hat{N}^j_{eff}$ is high enough, no resampling is necessary. Noting that the first-pass approximation in (18) is in fact the importance sampling density used for the APP method in (13), the absence of the target-resampling stage makes the sampling procedure equivalent to the one in APP, thus in these cases, only a reindexing step is necessary, setting $\mathbf{x}^{k+1}_{j,i} = \mathbf{x}^*_{j,a^i_j}$, for $i = 1, 2, ..., N$.

In order to provide a compact expression for the importance sampling function of ATRAPP, it is useful to make use of the set of target indices in the scenario at time $k+1$, $S^{k+1}_t = \{1, 2, ..., t\}$, and define a partition over S^{k+1}_t as

$$S^{k+1}_t = S^{k+1}_{APP} \cup S^{k+1}_{TRAPP}$$

$$\begin{aligned} \hat{N}^j_{eff} \leq \Gamma^{k+1}_j \quad &\Rightarrow \quad j \in S^{k+1}_{TRAPP} \\ \hat{N}^j_{eff} > \Gamma^{k+1}_j \quad &\Rightarrow \quad j \in S^{k+1}_{APP} \end{aligned}$$

with Γ^{k+1}_j, a parameter of ATRAPP that represents the target-resampling threshold for the j-th target at time step $k+1$. The

importance sampling function for ATRAPP can be therefore written as

$$q^{AT}(\mathbf{X}^{k+1}, \mathbf{a}|\mathbf{z}^{1:k+1}) = \prod_{j \in S^{k+1}_{TRAPP}} q^{T}_{j}(\mathbf{x}^{k+1}_{j}, a_j|\mathbf{z}^{1:k+1}) \prod_{j \in S^{k+1}_{APP}} q^{A}_{j}(\mathbf{x}^{k+1}_{j}, a_j|\mathbf{z}^{1:k+1}) \tag{21}$$

and introducing (21) into (7) finally yields the weight update equation for ATRAPP

$$\omega^{k+1}_{i} \propto \frac{p(\mathbf{z}^{k+1}|\mathbf{X}^{k+1}_{i})}{\prod_{j \in S^{k+1}_{TRAPP}} b_j(\mathbf{x}^{k+1}_{j,i}) \prod_{j \in S^{k+1}_{APP}} b_j(\boldsymbol{\mu}^{k+1}_{j,a^{i}_{j}})} \tag{22}$$

There are 2^t possible configurations for $q^{AT}(\mathbf{X}^{k+1}, \mathbf{a}|\mathbf{z}^{1:k+1})$, i.e. all possible combinations of choices for each target between $q^{T}_{j}(\mathbf{x}^{k+1}_{j}, a_j|\mathbf{z}^{1:k+1})$ and $q^{A}_{j}(\mathbf{x}^{k+1}_{j}, a_j|\mathbf{z}^{1:k+1})$. Selecting the optimal configuration for the importance density of ATRAPP at each time step is not straightforward. It is for this reason that we rely on a robust and well established criterion as the monitoring of the effective sample size for this purpose.

It should also be noted that the importance sampling function (21) of the proposed algorithm can accommodate a different resampling threshold for each target at each time step, Γ^{k}_{j}. This flexibility in Γ^{k}_{j} can serve different purposes, e.g., tuning the threshold for the possibly different dynamic models of each target, or for the possibly different measurement models of the sensors that surround each target.

In addition, the use of Γ^{k}_{j} also allows for the adequate accommodation of multidimensionality. As previously stated, the curse of dimensionality [27] naturally arises in the MTT problem. The difficulty of sampling a multitarget state posterior becomes higher when the number of targets grows. To tackle this problem, a different target-resampling threshold can be considered depending on the number of targets present in the surveillance area. This is in accordance with the mentioned insights into target-resampling exposed in Section IV. The higher the dimension, the more benefits we expect to obtain from target-resampling. Therefore, in general, the threshold Γ^{k}_{j} should be higher for a higher number of targets. A particular way of selecting this threshold policy is given in Section VI-C2. Finally, a pseudocode for the ATRAPP method is given through Algorithms 3 and 4.

Algorithm 3 ATRAPP prediction and update at time $k+1$

procedure $(\mathbf{z}^{k+1}, \{(\mathbf{X}^{k}_{1}, \omega^{k}_{1}), ..., (\mathbf{X}^{k}_{N}, \omega^{k}_{N})\})$

  for $i = 1, ..., N$ **do**
    for $j = 1, ..., t$ **do**
      $\mathbf{x}^{k+1|k}_{j,i} = \mathrm{E}[\mathbf{x}^{k+1}_{j,i}|\mathbf{x}^{k}_{j,i}]$
    end for
    $\hat{\mathbf{x}}^{k+1}_{j} = \sum_{i=1}^{N} \omega^{k}_{i} \cdot \mathbf{x}^{k+1|k}_{j,i}$
  end for
  for $j = 1, ..., t$ **do**
    $\{(\mathbf{x}^{k+1}_{j,1}, b_{j,1}), ..., (\mathbf{x}^{k+1}_{j,N}, b_{j,N})\} =$
      $target\_AT(\mathbf{z}^{k+1}, \hat{\mathbf{X}}^{k+1}_{-\{j\}}, \{(\mathbf{x}^{k}_{j,1}, \omega^{k}_{1}), ..., (\mathbf{x}^{k}_{j,N}, \omega^{k}_{N})\})$
  end for
  for $i = 1, ..., N$ **do** ▷ **Weight Update**
    - Compute $\omega^{k+1}_{i} = \frac{p(\mathbf{z}^{k+1}|\mathbf{X}^{k+1}_{i})}{\prod_{j=1}^{t} b_{j,i}}$
  end for
  for $i = 1, ..., N$ **do**
    - Normalize $\omega^{k+1}_{i} = \frac{\omega^{k+1}_{i}}{\sum_{i=1}^{N} \omega^{k+1}_{i}}$
  end for

  return $\{(\mathbf{X}^{k+1}_{1}, \omega^{k+1}_{1}), ..., (\mathbf{X}^{k+1}_{N}, \omega^{k+1}_{N})\}$

end procedure

VI. Experimental results

In this section, the performance of ATRAPP is tested via a set of simulations with a demanding MTT scenario. The proposed scenario is presented along with the employed motion and measurement models. It is shown how ATRAPP can respectively match the performances of both APP and TRAPP. Simulations also indicate that ATRAPP can in fact outperform both filters with an adequate selection of the threshold parameter, as well as other PFs for MTT in the literature.

The tracking performance of all PFs is tested by computing their OSPA [32] position error (with parameters $p = 2, \quad c =$ 10m) averaged over all targets, time steps of the trajectories and realizations in a Monte Carlo simulation with $n = 5000$ runs.

A. Motion modeling

The state of the j-th target is represented by the state vector $\mathbf{x}^{k}_{j} = [x^{k}_{j}, \dot{x}^{k}_{j}, y^{k}_{j}, \dot{y}^{k}_{j}]^{T}$, thus considering the target position and velocity. The motion of each target has been modeled as linear with a nearly constant velocity [34], so according to Assumption B

$$p(\mathbf{X}^{k+1}|\mathbf{X}^{k}) = \prod_{j=1}^{t} N(\mathbf{x}^{k+1}_{j}; \mathbf{F}\mathbf{x}^{k}_{j}, \mathbf{Q}) \tag{23}$$

$$\mathbf{F} = \mathbf{I}_2 \otimes \begin{pmatrix} 1 & \tau \\ 0 & 1 \end{pmatrix}$$

$$\mathbf{Q} = \sigma^{2}_{u} \mathbf{I}_2 \otimes \begin{pmatrix} \tau^3/3 & \tau^2/2 \\ \tau^2/2 & \tau \end{pmatrix}$$

where $N(\mathbf{x}; \bar{\mathbf{x}}, \mathbf{Q})$ is a Gaussian PDF with mean $\bar{\mathbf{x}}$ and covariance $\mathbf{Q}$ evaluated at $\mathbf{x}$, τ is the sampling period, $\mathbf{I}_n$ is the $n \times n$ identity matrix, $\otimes$ stands for the Kronecker product and σ^{2}_{u} is the continuous-time process noise intensity.

Simulations have been carried out with up to 8 targets present in the scenario whose trajectories repeatedly cross each other to recreate a demanding MTT problem. The trajectories have been generated according to equation (23), taking $\tau = 1s$ and $\sigma_u = 0.1m/s^{3/2}$. The simulated target trajectories have 100 time steps and are shown in Figure 2.

B. Sensor modeling.

The nonlinear measurement equation (3) for the i-th sensor at time $k+1$ is

$$z^{k+1}_{i} = h_i(\mathbf{X}^{k+1}) + v^{k+1}_{i} \tag{24}$$

Algorithm 4 target_AT subroutine

procedure $(\mathbf{z}^{k+1}, \hat{\mathbf{X}}^{k+1}_{-\{j\}}, \{(\mathbf{x}^k_{j,1}, \omega^k_1), ..., (\mathbf{x}^k_{j,N}, \omega^k_N)\})$

for $i = 1, ..., N$ **do** ▷ **Auxiliary filtering**
- Draw $\boldsymbol{\mu}^{k+1}_{j,i}$ using: $p(\mathbf{x}^{k+1}_j|\mathbf{x}^k_{j,i})$ or $\mathrm{E}[\mathbf{x}^{k+1}_j|\mathbf{x}^k_{j,i}]$
- Compute $b_j(\boldsymbol{\mu}^{k+1}_{j,i}) = p(\mathbf{z}^{k+1}|\hat{\mathbf{X}}^{k+1}_{-\{j\}}, \boldsymbol{\mu}^{k+1}_{j,i})$
- Compute $\lambda_{j,i} = b_j(\boldsymbol{\mu}^{k+1}_{j,i})\omega^k_i$

end for
for $i = 1, ..., N$ **do**
- Normalize $\lambda_{j,i} = \frac{\lambda_{j,i}}{\sum_{i=1}^N \lambda_{j,i}}$

end for
for $i = 1, ..., N$ **do**
- Sample an index a^i_j from the distribution defined by $(\lambda_{j,1}, \lambda_{j,2}, ..., \lambda_{j,N})$
- Draw a sample $\mathbf{x}^*_{j,a^i_j}$ from $p(\mathbf{x}^{k+1}_j|\mathbf{x}^k_{j,a^i_j})$

end for

for $i = 1, ..., N$ **do**
- Compute $b_j(\mathbf{x}^*_{j,a^i_j}) = p(\mathbf{z}^{k+1}|\hat{\mathbf{X}}^{k+1}_{-\{j\}}, \mathbf{x}^*_{j,a^i_j})$
- Compute $r_j(\mathbf{x}^*_{j,a^i_j}) = \frac{b_j(\mathbf{x}^*_{j,a^i_j})}{b_j(\boldsymbol{\mu}^{k+1}_{j,a^i_j})}$

end for
for $i = 1, ..., N$ **do**
- Normalize $r_j(\mathbf{x}^*_{j,a^i_j}) = \frac{r_j(\mathbf{x}^*_{j,a^i_j})}{\sum_{i=1}^N r_j(\mathbf{x}^*_{j,a^i_j})}$

end for
- Compute $\hat{N}^j_{eff} = \frac{1}{\sum_{i=1}^N (r_j(\mathbf{x}^*_{j,a^i_j}))^2}$

if $\hat{N}^j_{eff} > \Gamma^{k+1}_j$ **then**
for $i = 1, ..., N$ **do**
- Set $\mathbf{x}^{k+1}_{j,i} = \mathbf{x}^*_{j,a^i_j}$
- Set $b_{j,i} = b_j(\boldsymbol{\mu}^{k+1}_{j,i})$

end for
else
for $i = 1, ..., N$ **do** ▷ **target-resampling**
- Sample an index p from the distribution defined by $(r_j(\mathbf{x}^*_{j,a^1_j}), r_j(\mathbf{x}^*_{j,a^2_j}), ..., r_j(\mathbf{x}^*_{j,a^N_j}))$
- Set $\mathbf{x}^{k+1}_{j,i} = \mathbf{x}^*_{j,a^p_j}$
- Set $b_{j,i} = b_j(\mathbf{x}^*_{j,a^p_j})$

end for
end if

return $\{(\mathbf{x}^{k+1}_{j,1}, b_{j,1}), ..., (\mathbf{x}^{k+1}_{j,N}, b_{j,N})\}$

end procedure

$$h_i(\mathbf{X}^{k+1}) = \sum_{j=1}^{t} SNR(d^{k+1}_{j,i})$$

$$SNR(d^{k+1}_{j,i}) = \begin{cases} SNR_0 & d^{k+1}_{j,i} \leq d_0 \\ SNR_0 \frac{d_0^2}{(d^{k+1}_{j,i})^2} & d^{k+1}_{j,i} > d_0 \end{cases}$$

$$d^{k+1}_{j,i} = \sqrt{(x^{k+1}_j - \mathrm{s}_{x,i})^2 + (y^{k+1}_j - \mathrm{s}_{y,i})^2}$$

where v^{k+1} is a zero-mean, unit-variance, Gaussian-distributed noise. SNR_0 is the maximum signal to noise ratio produced by a target when it is closer to the sensor than the saturation distance, d_0. The coordinates of the i-th sensor are given by its position vector, $\boldsymbol{s}_i = [\mathrm{s}_{x,i}, \mathrm{s}_{y,i}]^T$. In the simulated scenario, measurements are taken from a set of sensors displayed in a flat squared regular grid covering $120\mathrm{m}\times120\mathrm{m}$, with a spacing of $10\mathrm{m}$ between sensors in both axes, yielding a total of 169 sensors, SNR_0 has been taken to be $20\mathrm{dB}$ and $d_0 = 5\mathrm{m}$.

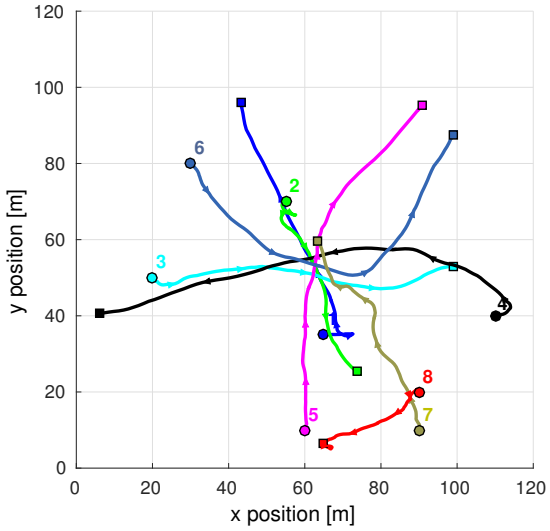


Figure 2: Eight simulated trajectories, consisting of 100 steps sampled with a period of $\tau = 1s$. The initial point of each trajectory is indicated by a colored circle, while the final point is represented by a colored square. An arrow shows the position of the target at steps 20, 40, 60 and 80 of each trajectory. When simulating a scenario with t targets, target trajectories 1 to t of the above presented are taken into account.

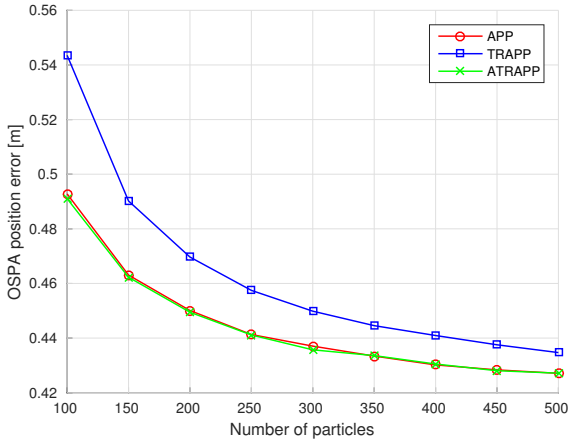
Sensor measurements do not depend on the velocity of the targets, as they are only affected by their position. As the relationship between the velocity and the position of the targets is linear, Rao-Blackwellization is applied to reduce the curse of dimensionality [35][4], only including the position of the targets in the PF and optimally estimating their velocities by Kalman filtering.

C. Simulation results

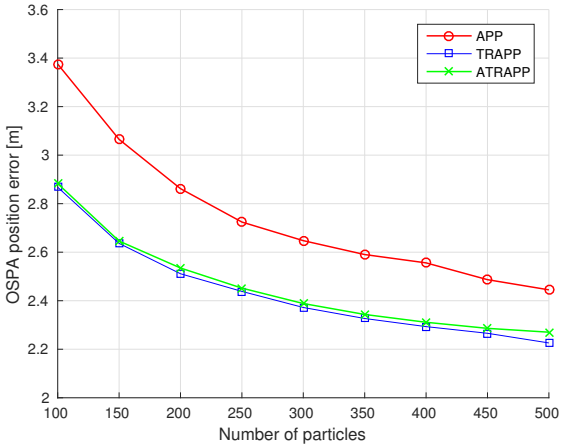
1) Constant threshold: We first analyze the capabilities of ATRAPP using a simple target-resampling threshold policy where Γ^k_j is set constant, thus $\Gamma^k_j = \Gamma$. This approach has some advantages, first, it can be used to analyze the equivalence of the ATRAPP PF with respect to APP and TRAPP PFs when $\Gamma = 0$ and $\Gamma = N$, respectively. In addition, this constant threshold policy has the advantage of needing the adjustment of only one parameter, which simplifies the filter design. This parameter is set in advance so that no additional computational expense is required.

In Figure 3 we show the error results for 2 and 8 targets. It can be seen that the better performance between APP and TRAPP switch when considering 2 targets in Figure 3(a) with respect to the results obtained when considering 8 targets in the scenario in Figure 3(b). This is in accordance with the insights explained in Section IV, showing that when the dimensionality of the state space grows, the target-resampling stage is advised.

It would be desirable that a unique policy in ATRAPP served to reach the performances of both APP and TRAPP PFs regardless of the situation. This may not be generally

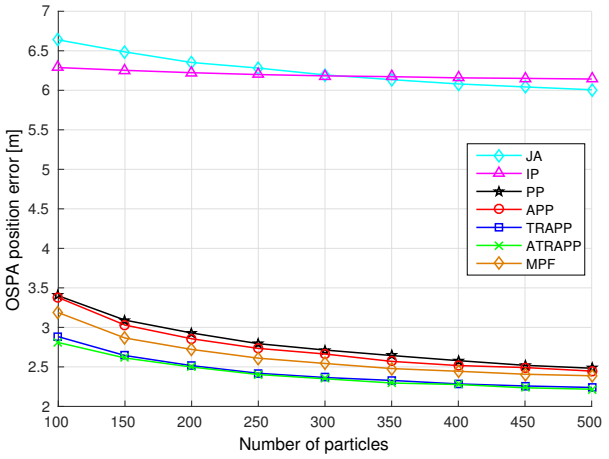


(a)

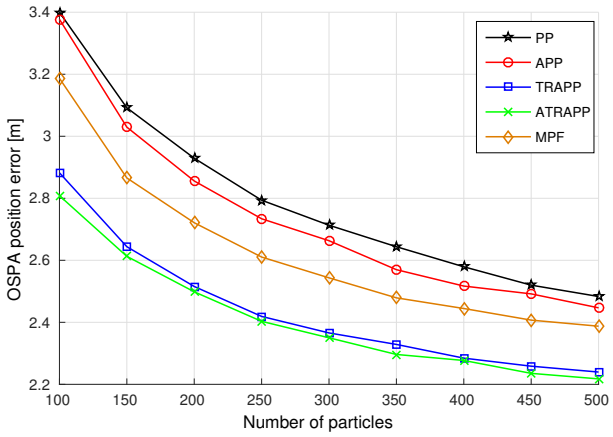


(b)

Figure 3: OSPA position error with respect to the number of particles with 3 different filters: APP, TRAPP and ATRAPP with $\Gamma = 0.5 \cdot N$. Results show how this parametrization of ATRAPP can match the best performance of APP when 2 targets (3a) are present in the scenario as well as the best performance of TRAPP when 8 targets (3b) are present in the scenario.



(a)



(b)

Figure 4: OSPA position error with respect to the number of particles when 8 targets are under track with different filters; JA, IP, PP, MPF, APP, TRAPP, ATRAPP with $\Gamma = 0.8 \cdot N$. (4b) only shows the results of the 5 latter algorithms.

attainable, however, a fine parametrization of the thresholds in ATRAPP can bring this objective closer. Figure 3 illustrates how in the proposed scenario, a target-resampling threshold of $\Gamma = 0.5 \cdot N$ makes ATRAPP to match the performance of APP when there is not a high number of targets in the scenario and match the performance of TRAPP when this number of targets grows. In this case, simulations indicate that ATRAPP can handle the situation of sampling the state space for 2 targets with their effective target sample sizes not falling below the 50% of the number of particles, thus not usually performing target-resampling and matching the performance of APP. On the other hand, results show that the effective target sample size tends to fall below the 50% of the number of particles when sampling the 8-target state space, thus making ATRAPP prone to target-resampling in this situation and allowing it to match the best performance of TRAPP in such a complex scenario.

It has been shown how a simple policy of a midpoint threshold selection allows ATRAPP to match the performance of APP and TRAPP when they respectively accomplish best results. However, a fine choice of the target threshold parameter should make ATRAPP to apply target-resampling only when necessary to favor particle diversity. Figure 4 shows how ATRAPP can, not only match the performances of APP and TRAPP, but also marginally outperform either of them in the considered scenario with a fine choice of the thresholding parameter.

Note that if we compare ATRAPP with APP and TRAPP considering both previous scenarios simultaneously, ATRAPP outperforms both of them in general as it significantly outperforms APP or TRAPP in one of the scenarios, TRAPP when there are two targets and APP when there are eight targets.

In the case of Figure 4, a constant target threshold of $\Gamma = 0.8 \cdot N$ is adopted, resulting in the mentioned improvement of the performance of ATRAPP with respect to APP and TRAPP.

Figure 4 also shows the comparative results with respect to other MTT PFs: IP [15], PP [8], the MPF [17], [18], [19], and the jointly auxiliary (JA) PF, which is a version of the

traditional APF that suits the MTT problem as in [36]. This filter jointly samples the whole multitarget state space, therefore it accounts for nearby targets in the sampling procedure, but does not perform subparticle crossover. It can be observed how both JA and IP show remarkably more error; in the case of JA this is caused by the curse of dimensionality arising due to the joint sampling of the multitarget state space. On the other hand, as IP does not take into account neighboring targets in the importance density, its performance plummets in this scenario with a high number of target crossings. The MPF, while providing low estimation errors, does not approximate the joint posterior PDF over the multiple targets, which is quite useful in some situations [20]. It is also worth mentioning that although PP does not acutely suffer from the curse of dimensionality nor from the presence of neighboring targets, it is clearly outperformed by APP and TRAPP. In addition PP is also outperformed by ATRAPP due to the adaptive target-resampling strategy.

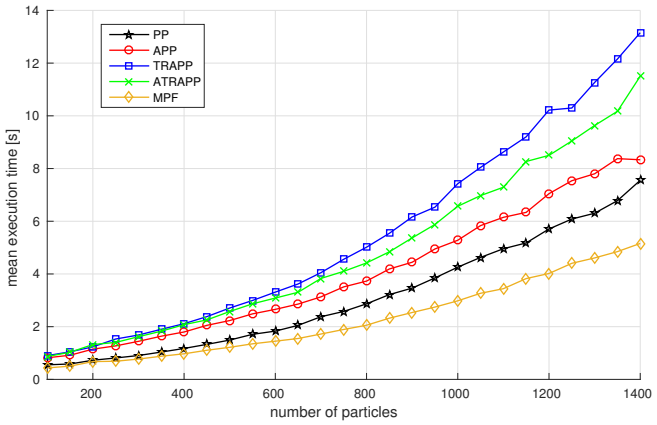
As the methods considered in this section have different computational loads for the same number of particles, it is also worth analyzing the performance of PFs for a given computational load, which is measured by the execution time. As a previous remark, notice that the results in this kind of analysis can appreciably vary depending on the implementation of the algorithms and the computing platform, among other factors. The execution times of the Matlab implementation of the considered algorithms on an Intel Core i7 computer are presented in Figure 5(a).

It is shown in 5(b) how if hard real-time constraints are present, the best option in terms of performance for small execution times is to use those algorithms with very low computational burden, in particular, MPF. However, as previously mentioned, this method approximates individual target states by their marginal PDFs, so its application is restricted if the joint PDF of the multitarget state is required [20]. When higher execution times are affordable, ATRAPP is clearly the best choice, as it not only outperforms all the filters that estimate the joint multitarget state PDF, but also outperforms MPF.

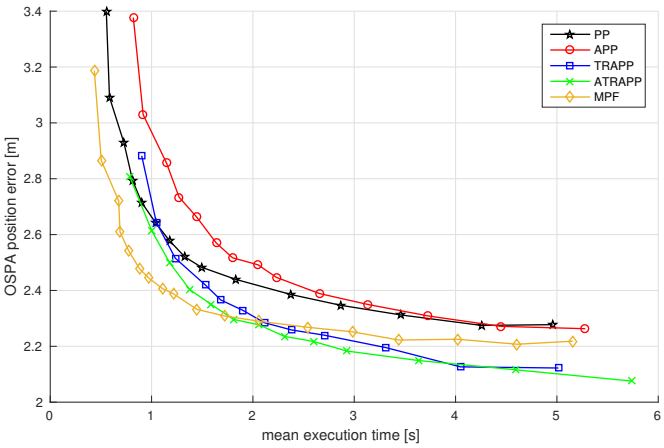
2) Variable threshold: As previously stated, the optimal thresholding policy for ATRAPP can result in a great improvement in the filter performance, as it can account for the best possible target-resampling strategy. This policy should take into account the different dynamics of each target, the measurement model of the surrounding sensors and the number of targets to deal with the dimensionality problems arising in MTT.

In this section, we define a sample target-resampling threshold selection policy based on the number of targets in the scenario. The higher the number of targets, the higher the dimension and, therefore, the better target-resampling is expected to perform, as indicated in Section IV. An example of a policy that favors target-resampling for high target number is shown in Figure 6.

Figure 7 shows the results of the applied policy for the proposed simulation scenario when the number of targets being tracked varies from $t = 2$ to $t = 8$. It is shown how ATRAPP either matches or improves the performance of APP and TRAPP PFs, a clearer view of this improvement is



(a)



(b)

Figure 5: (a): Mean execution times with respect to the number of particles for the different algorithms when $t = 8$ targets are present at the scenario. Execution times consider the estimation of the 100 steps of the trajectories. (b): OSPA position error with respect to the mean execution time of the different algorithms.

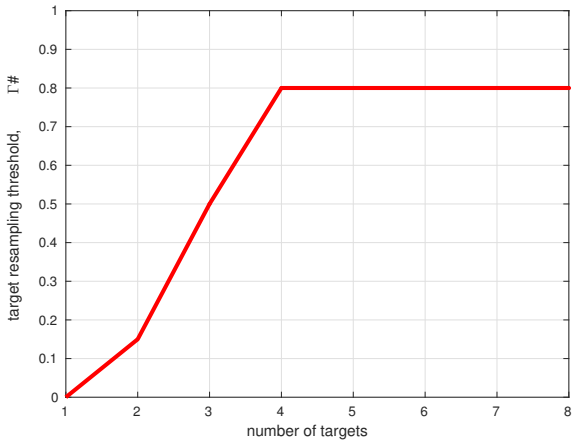
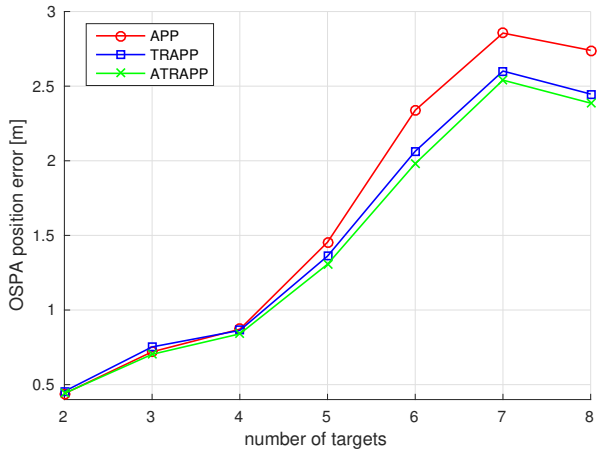
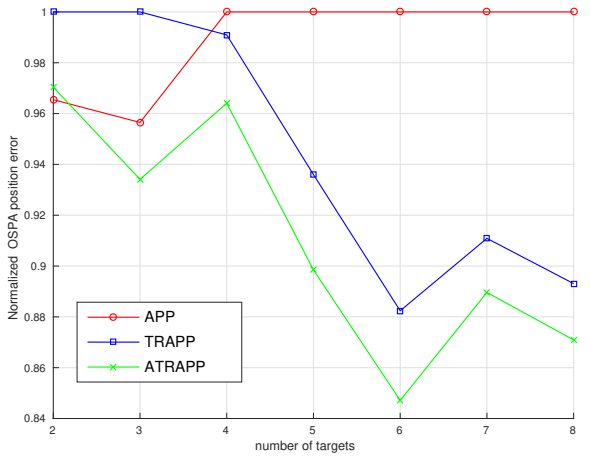


Figure 6: Target-resampling threshold selection policy dependent on the number of targets being tracked. The threshold is normalized with respect to the total number of particles. The upper bound of $\Gamma = 0.8 \cdot N$ is based on the best results obtained in previous simulations.

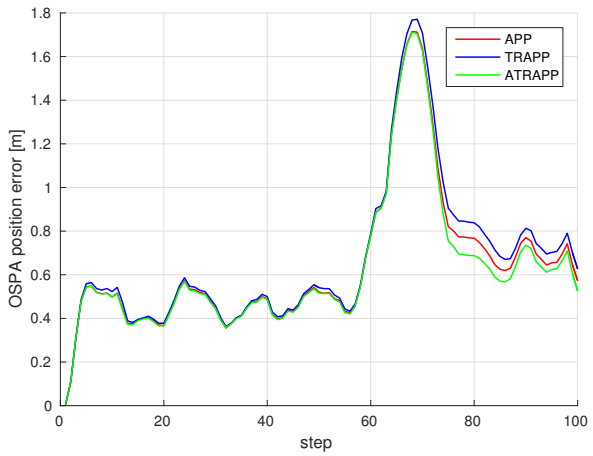


(a)

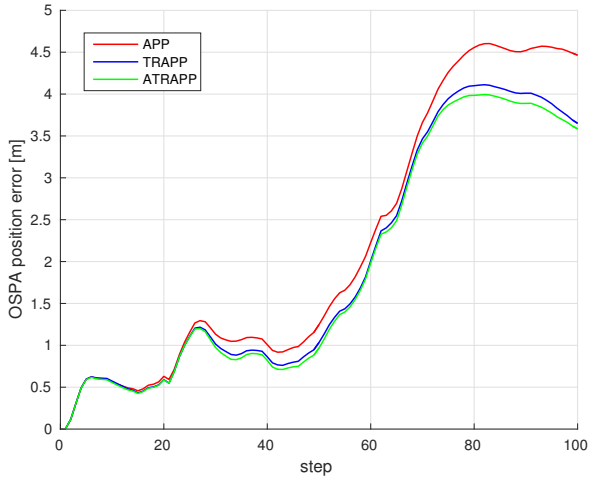


(b)

Figure 7: Normalized OSPA position error with respect to the worst performing algorithm using APP, TRAPP and ATRAPP PFs with $N = 250$ particles when 2 to 8 targets are present in the scenario. ATRAPP uses the target-resampling threshold selection policy in Figure 6. (7a) shows the OSPA position error and (7b) the improvement with respect to the worst performing filter in each case.



(a)



(b)

Figure 8: OSPA position error at each time step of the trajectories for a scenario with 3 targets (8a) and 8 targets (8b) for APP, TRAPP and ATRAPP PFs with $N = 250$ particles. ATRAPP uses the target-resampling threshold selection policy in Figure 6.

shown in Figure 7(b), where the errors between APP, TRAPP and ATRAPP have been normalized with respect to that of the worst performing filter in each simulation, showing how ATRAPP either matches or outperforms APP and TRAPP.

The origin of the improvement in ATRAPP is explained in Figure 8. It is shown how all the filters increase their error when target trajectories cross (almost all crossings approximately start at time step 50, see Figure 2). Nevertheless, it is shown how ATRAPP is the filter which better reacts to these difficult situations by performing a faster and more successful recovery, achieving this by favoring subparticle diversity when possible and performing target-resampling only when it is really needed to avoid subparticle degeneracy.

In this paper, we have derived ATRAPP for a fixed and known number of targets. We would like to mention that we can also use this method for sampling the surviving targets in a general MTT problem, which includes a varying and unknown number of targets, as in the algorithm presented in [8]. Conditioning on the surviving targets at each time step, we can use ATRAPP, instead of PP as proposed in [8]. In this context, the use of the proposed policy in Figure 6 would adequately respond to the necessities of the filter as the number of targets in the scenario changes.

VII. Conclusions

In this paper, we have extensively presented APP and TRAPP particle filters for multiple target tracking. We have also illustrated and analyzed their behavior with an increasing number of targets. In order to combine the strengths of both filters, we have also proposed ATRAPP.

ATRAPP makes use of state partitioning, auxiliary filtering and an adaptive target-resampling scheme to be able to attain good performance in difficult, high dimensional track-before-detect scenarios. Making use of extensive numerical

simulations, we have also demonstrated that ATRAPP does not only outperform both APP and TRAPP in different MTT scenarios but also other previous known methods for MTT in the literature.

We have also demonstrated how simple resampling threshold policies can make ATRAPP achieve remarkable results in a demanding MTT scenario. Future work includes the search of an optimum target-resampling threshold policy.

REFERENCES

[1] Särkkä, S., *Bayesian Filtering and Smoothing*. Cambridge University Press, 2013.

[2] Bar-Shalom, Y., *Multitarget-multisensor tracking: advanced applications*. Artech House, 2000.

[3] Mahler, R. P. S., *Statistical Multisource-Multitarget Information Fusion*. Norwood, MA, USA: Artech House, Inc., 2007.

[4] Morelande, M. R., Kreucher, C. M., and Kastella, K., "A Bayesian approach to multiple target detection and tracking," *IEEE Transactions on Signal Processing*, vol. 55, pp. 1589–1604, May 2007.

[5] Ekman, M., Sviestins, E., and Systems, S., "Particle Filters for Tracking Closely Spaced Targets," *Proceedings of the10th International Conference on Information Fusion (FUSION)*, 2010.

[6] García-Fernández, Á. F., Morelande, M. R., and Grajal, J., "Bayesian Sequential Track Formation," *IEEE Transactions on Signal Processing*, vol. 62, no. 24, pp. 6366–6379, 2014.

[7] García-Fernández, Á. F. and Morelande, M. R., "Explicit filtering equations for labelled random finite sets," *International Conference on Control, Automation and Information Sciences (ICCAIS)*, pp. 349–354, 2015.

[8] García-Fernández, Á. F., Grajal, J., and Morelande, M., "Two-layer particle filter for multiple target detection and tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 3, pp. 1569–1588, 2013.

[9] García-Fernández, Á. F., "Track-Before-Detect Labeled Multi-Bernoulli Particle Filter With Label Switching," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 5, pp. 2123–2138, 2016.

[10] Daum, F. E., Company, R., and Division, E., "New exact nonlinear filters: theory and applications," *Proc. SPIE 2235, Signal and Data Processing of Small Targets*, vol. 2235, pp. 636–649, 1994.

[11] Crisan, D. and Doucet, A., "A survey of convergence results on particle filtering methods for practitioners," *IEEE Transactions on Signal Processing*, vol. 50, no. 3, pp. 736–746, 2002.

[12] Hu, X. L., Schön, T. B., and Ljung, L., "A basic convergence result for particle filtering," *IEEE Transactions on Signal Processing*, vol. 56, no. 4, pp. 1337–1348, 2008.

[13] Hu, X. L., Schön, T. B., and Ljung, L., "A General Convergence Result for Particle Filtering," *IEEE Transactions on Signal Processing*, vol. 59, no. 7, pp. 3424–3429, 2011.

[14] Yi, W., Morelande, M. R., Kong, L., and Yang, J., "A computationally efficient particle filter for multitarget tracking using an independence approximation," *IEEE Transactions on Signal Processing*, vol. 61, pp. 843–856, Feb 2013.

[15] Kreucher, C. M., Kastella, K., and O. Hero III, A., "Multitarget tracking using the joint multitarget probability density," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 41, no. 4, pp. 1396–1414, 2005.

[16] Maskell, S., Rollason, M., Gordon, N., and Salmond, D., "Efficient particle filtering for multiple target tracking with application to tracking in structured images," *Image and Vision Computing*, vol. 21, no. 10, pp. 931–939, 2003.

[17] Bugallo, M. F., Lu, T., and Djuric, P. M., "Target Tracking by Multiple Particle Filtering," *IEEE Aerospace Conference*, pp. 1–7, 2007.

[18] Djurić, P. M., Ting, L., and Bugallo, M. F., "Multiple particle filtering," *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, vol. 3, pp. 1181–1184, 2007.

[19] Beaudeau, J. P., Bugallo, M. F., and Djurić, P. M., "RSSI-Based Multi-Target Tracking by Cooperative Agents Using Fusion of Cross-Target Information," *IEEE Transactions on Signal Processing*, vol. 63, no. 19, pp. 5033–5044, 2015.

[20] García-Fernández, Á. F., Vo, B.-N., and Vo, B.-T., "MCMC-based posterior independence approximation for RFS multitarget particle filters," in *Proceedings of the17th International Conference on Information Fusion (FUSION)*, 2014.

[21] Ristic, B., Arulampalam, S., and Gordon, N., *Beyond the Kalman filter, particle filters for tracking applications*. Artech House, 2004.

[22] Doucet, A., Godsill, S., and Andrieu, C., "On sequential Monte Carlo sampling methods for Bayesian filtering," *Statistics and Computing*, vol. 10, no. 3, pp. 197–208, 2000.

[23] Pitt, M. K., Shephard, N., and Gate, Q., "Filtering via simulation: Auxiliary particle filters," *Journal of the American Statistical Association*, pp. 590–599, 1999.

[24] Úbeda-Medina, L., García-Fernández, Á. F., and Grajal, J., "Generalizations of the auxiliary particle filter for multiple target tracking," in *Proceedings of the17th International Conference on Information Fusion (FUSION)*, 2014.

[25] Arulampalam, M., Maskell, S., Gordon, N., and Clapp, T., "A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[26] Oh, M.-S., "Monte Carlo integration via importance sampling: dimensionality effect and an adaptive algorithm," *Contemporary Mathematics*, vol. 115, no. 1991, pp. 165–187, 1991.

[27] Daum, F. and Huang, J., "Curse of dimensionality and particle filters," *Proceedings of 2003 IEEE Aerospace Conference*, vol. 4, pp. 1979–1993, 2003.

[28] García-Fernández, Á. F., *Detection and tracking of multiple targets using wireless sensor networks*. Ph. d. dissertation, Universidad Politécnica de Madrid, 2011.

[29] B. Rubin, D., "A Noniterative Sampling/Importance resampling alternative to data augmentation for creating a few imputations when fractions of missing information are modest: The SIR algorithm," *Journal of the American Statistical Association*, vol. 82, pp. 544–546, 1987.

[30] Orton, M. and Fitzgerald, W., "A Bayesian approach to tracking multiple targets using sensor arrays and particle filters," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 216–223, 2002.

[31] Klaas, M., de Freitas, N., and Doucet, A., "Toward Practical N^2 Monte Carlo: The Marginal Particle Filter," *Proceedings of the Twenty-First Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pp. 308–315, 2005.

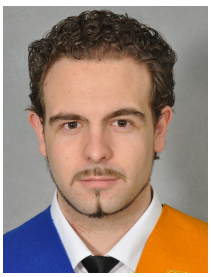
[32] Schuhmacher, D., Vo, B.-T., and Vo, B.-N., "A consistent metric for performance evaluation of multi-object filters," *IEEE Transactions on Signal Processing*, vol. 56, pp. 3447–3457, Aug 2008.

[33] Kong, A., Liu, J. S., and Wong, W. H., "Sequential imputations and Bayesian missing data problems," *Journal of the American statistical association*, vol. 89, no. 425, pp. 278–288, 1994.

[34] Bar-Shalom, Y., Li, X.-R., and Kirubarajan, T., *Estimation with applications to tracking and navigation*. John Wiley & Sons, Inc., 2001.

[35] Schön, T., Gustafsson, F., and Nordlund, P.-J., "Marginalized particle filters for mixed linear/nonlinear state-space models," *IEEE Transactions on Signal Processing*, vol. 53, pp. 2279–2289, Jul 2005.

[36] Morelande, M. R., "Tracking multiple targets with a sensor network," in *Proceedings of the 9th International Conference on Information Fusion (FUSION)*, 2006.



Luis Úbeda-Medina received the B. Sc. and M. Sc. degree in Telecommunication Engineering from the Technical University of Madrid, Spain, in 2012. Since 2012, he has been working towards the Ph.D. degree at the Microwave and Radar Group of the Technical University of Madrid. His research activities and interests include control theory and Bayesian inference in dynamic systems, radar signal processing and 3D image processing.

Ángel F. García Fernández received the telecommunication engineering degree (with honours) and the Ph.D. degree from the Technical University of Madrid, Madrid, Spain, in 2007 and 2011, respectively.

He has held postdoctoral positions at the Technical University of Madrid, Chalmers University of Technology, Gothenburg, Sweden, and Curtin University, Perth, Australia. He is currently a Postdoctoral Researcher in the Department of Electrical Engineering and Automation at Aalto University, Espoo, Finland. His main research activities and interests are in the area of Bayesian nonlinear inference, with emphasis on dynamic systems.

Jesús Grajal was born in Toral de los Guzmanes, Leon, Spain, in 1967. He received his Ingeniero de Telecomunicacion degree and the PhD degree from the Universidad Politécnica de Madrid (UPM), Spain, in 1992 and 1998, respectively. Since 2001 he has been an associate professor at the Signals, Systems, and Radiocommunications Department of the Universidad Politécnica de Madrid. His research activities are in the area of hardware-design for radar systems, radar signal processing, and broadband digital receivers for radar and spectrum surveillance applications.