Two linear complexity particle filters capable of maintaining target label probabilities for targets in close proximity

Two Linear Complexity Particle Filters Capable of Maintaining Target Label Probabilities for Targets in Close Proximity

Ramona Georgescu and Peter Willett
Electrical and Computer Engineering
University of Connecticut
Storrs, CT 06269
Email: {ramona,willett}@engr.uconn.edu

Lennart Svensson
Signals and Systems
Chalmers University of Technology
Gothenburg, Sweden
Email: lennart.svensson@chalmers.se

Mark Morelande
Electrical and Electronic Engineering
University of Melbourne
Victoria, Australia
Email: m.morelande@ee.unimelb.edu.au

***Abstract*—In this work, we introduce two particle filters of linear complexity in the number of particles that take distinct approaches to solving the problem of tracking two targets in close proximity. We operate in the regime in which measurements do not discriminate between targets and hence uncertainties in the labeling of the tracks arise. For simplicity, we limit our study to the two target case for which there are only two possible associations between targets and tracks. The proposed Approximate Set Particle Filter (ASPF) introduces some approximations but has similar complexity and still provides much more accurate descriptions of the posterior uncertainties compared to standard particle filters. The fast Forward Filter Unlabeled Backward Simulator (fast FFUBSi) employs a smoothing technique based on rejection sampling for the calculation of target label probabilities. Simulations show that neither particle filter suffers from track coalescence (when outputting MMOSPA estimates) and both calculate correct target label probabilities.**

Keywords: Particle filter, target labels, linear complexity.

I. Introduction

In theory, particle filters provide optimal solutions to any target tracking problem. However, in practice it has been observed that they have significant problems in situations when targets have been closely spaced for some time. The weaknesses are related to the particle filters' inability to accurately express uncertainties regarding past states; due to degeneracy, they always overestimate how certain they are about the past and going sufficiently far back (usually only a few time steps) they claim that they know the exact value of the state vector [1].

In a tracking scenario where two targets are closely spaced for some time and later resolve, there are uncertainties as to which target is which, i.e., what we refer to as target labeling uncertainties. Based on the information obtained from, e.g., a radar sensor, it is typically not possible to resolve uncertainties in labeling once they have appeared. Still, due to degeneracy, which is an inherent part of any particle filter (with finite number of particles), the filter will soon claim that it knows the correct labeling with unity probability [2]. Clearly, this does not reflect the true posterior distribution and moreover, it may as applied have significant repercussions. Consequently, recent years have brought significant interest in developing particle filters able to maintain accurate target label probabilities.

In [3], Blom and Bloem propose a new particle filter able to provide an estimated track swap probability for the case of two closely spaced linear Gaussian targets. At its core lies a unique decomposition of the joint conditional density as a mixture of a permutatixon invariant density and a permutation strictly variant density. While the ASPF takes a similar approach, our fast FFUBSi employs a very different (smoothing) technique.

In [4], Garcia-Fernandez et al. introduce a novel particle filter able to maintain the multimodality of the posterior pdf after the targets have moved in close proximity and thus, to extract information about target labels. The drawback is that complexity grows as $O(M^2)$, where M is the number of particles, due to the association of a probability vector to each particle. Instead, our two approaches associate the same vector to all the particles, which can be done with linear complexity.

In this work, the first type of particle filter we introduce associates a probability vector to each particle. The probability vector represents the probabilities for different labeling events. For instance, if the targets have been well separated at all times, one of the elements in the vector will be one and other(s) will be zero. If, on the other hand, the targets have been very closely spaced for a long time all elements in the vector are the same. Note that there are only two possible labelings for the two target case and therefore the length of the probability vector is two. But since the elements of the vector have to sum to one, we really only have one free parameter. We call this new filter the Approximate Set Particle Filter (ASPF).

The second type of particle filter we introduce takes an approach of forward filtering followed by backward smoothing. Target labels produced by the forward filter are ignored and target identity probabilities are calculated based solely on the backward trajectories generated by performing rejection sampling backwards in time. We call this new filter the fast Forward Filter Unlabeled Backward Simulator (fast FFUBSi).

The paper is organized as follows. Section II formulates the target tracking problem. Sections III and IV describe the two novel particle filters with the ability of extracting target

label probabilities. In Section V, we show results with the new particle filters operating on simulated scenarios and then conclude in Section VI.

II. Problem Statement

Multitarget tracking algorithms represent the multitarget state either as sets or as vectors. The former approach implies that one is not interested to distinguish between the targets and the sole purpose is estimation of their states. On the other hand, the latter approach is concerned with knowing which target is which in addition to estimating their states. The information regarding target labeling is available in and can be extracted from the posterior pdf of the multitarget state [4].

Suppose we wish to track two targets[1] with state vectors x_k^1 and x_k^2, collected in a joint state vector

$$x_k = \begin{bmatrix} x_k^1 \\ x_k^2 \end{bmatrix}. \tag{1}$$

The two targets are assumed to move independently of each other and to follow the same motion model

$$x_k^j = f_k(x_{k-1}) + v_{k-1}^j, \tag{2}$$

with process noise $v_{k-1}^j \sim \mathcal{N}(\mathbf{0}, Q_k)$ and $j \in \{1, 2\}$.

The measurement data observed at time k is denoted z_k and the collection of data up to and including time k is

$$Z^k = \left[z_1,\ z_2,\ \dots z_k\right]. \tag{3}$$

The data is assumed to follow the common assumptions for radar sensor data [5]. A solution to this tracking problem has to handle the usual data association uncertainties. Among other things, these uncertainties imply that the measurement model will be such that

$$h(z_k|x_k) = h(z_k|\chi x_k), \tag{4}$$

where χ is the permutation matrix

$$\chi = \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}. \tag{5}$$

and I is the identity matrix.

Assuming the posterior pdf at time k is available, the predicted pdf at time $k+1$ is calculated using Eq. (2). The posterior pdf at time $k+1$ is obtained by Bayesian update [6] using the predicted pdf at time $k+1$ and the received measurements.

Particle filters (PFs) are commonly used to approximate the full Bayesian posterior pdf when the system is nonlinear and/or nonGaussian. Unfortunately, due to particle degeneracy, PFs are unable well to approximate posterior pdfs with multiple modes over longer periods of time. A traditional PF will not represent well such a posterior pdf and therefore would be unable to provide reliable information about target labels, information that lies in the multimodality of the posterior pdf.

Targets coming into close proximity is a commonly encountered scenario in which the posterior pdf is multimodal. In this paper, we propose two different algorithms that attempt to solve the (same) problem of building tracks for such a scenario while maintaining correspondence of target states over time.

[1]We will treat only two targets, with no loss of generality as regards ideas and considerably more compact notation than a greater set.

III. Approximate Set Particle Filter (ASPF)

In our first linear-complexity particle filter capable of maintaining target label probabilities, i.e., the Approximate Set Particle Filter (ASPF), we represent the posterior density of the tracks at each time scan using a set of M particles, $x_k^{(1)}, x_k^{(2)}, \dots, x_k^{(M)}$ to which we associate weights $w_k^{(1)}, w_k^{(2)}, \dots, w_k^{(M)}$. Additionally, we store the uncertainties in the target-to-track associations in a separate variable and we recursively update this variable. In the general n target case, the track-to-target probabilities are vectors of length $n!$ since that is the number of possible permutations of a state vector x_k. When we only have two targets, the vector contains two elements where the second element is completely determined by the first (the elements have to add up to one). In this work, we therefore find it sufficient to deal with the first element of that vector, here denoted p_k at each time scan.

The posterior density of the tracks is approximated as:

$$\pi(x_k|Z^k) \approx \sum_{i=1}^{M} w_k^{(i)} \delta(x_k - x_k^{(i)}), \tag{6}$$

which is identical to the one in a conventional particle filter. However, to distinguish tracks from targets, we introduce a separate density for the targets that we describe as,

$$\tilde{\pi}(x_k|Z^k) \approx p_k \pi(x_k|Z^k) + (1 - p_k)\pi(\chi x_k|Z^k). \tag{7}$$

We implicitly assume that the distribution of the targets can be expressed (at least approximately) as in Eq. (7), where the pdf of the tracks (in Eq. (6)) contains no labeling uncertainties. Specifically, we assume that there is a point $\hat{x}$ such that $\pi(x_k|Z^k) = 0$ for all x_k satisfying $||\hat{x} - x_k||^2 > ||\hat{x} - \chi x_k||^2$.

We now proceed to describe how to recursively update the particles, their weights and the probability, p_k.

A. ASPF Algorithm

The ASPF sequentially computes target label probabilities by approximating the MMOSPA[2] posterior pdf, $\tilde{\pi}_k(X)$ instead of the traditional labeled posterior pdf, $\pi_k(x)$.

Applying the Chapman-Kolmogorov-Bayes theorem gives:

$$\bar{\omega}_{k+1}(x) \propto L_k(x) \int f(x|x_k)\pi_k(x_k)dx_k \tag{8}$$

where L_k is the likelihood at time k and $f(\cdot|\cdot)$ is the multitarget state transition density. Note that $\bar{\omega}_{k+1}(x)$ should not be interpreted as the labeled posterior, but rather as a one-step labeled posterior.

The MMOSPA posterior pdf at time $k+1$ is defined as:

$$\tilde{\pi}_{k+1}(x) = \begin{cases} \bar{\omega}_{k+1}(x) + \bar{\omega}_{k+1}(\chi x) & \text{if } x \in A_{k+1}(\hat{x}_{k+1}) \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

[2]The MMOSPA estimator minimizes the mean Optimal Subpattern Assignment (OSPA) metric. The reader is encouraged to consult [7] for the derivation and the intuition behind MMOSPA estimation.

where $A_{k+1}(\hat{x}_{k+1}) = \{x : ||\hat{x}_{k+1} - x_{k+1}|| \leq ||\hat{x}_{k+1} - \chi x_{k+1}||\}$ and $\hat{x}_{k+1}$ is the MMOSPA estimator:

$$\hat{x}_{k+1} = \int_{A_{k+1}(\hat{x}_{k+1})} x \tilde{\pi}_{k+1}(x) dx \tag{10}$$

The probability that $\hat{x}_{k+1}$ has a labeling error, with respect to $\bar{\omega}_{k+1}(x)$, is:

$$\tilde{p}_{k+1} = \int_{\bar{A}_{k+1}(\hat{x}_{k+1})} \bar{\omega}_{k+1}(x) dx \tag{11}$$

The posterior target labeling probabilities can then be calculated as:

$$p_{k+1} = (1 - \tilde{p}_{k+1}) p_k + \tilde{p}_{k+1}(1 - p_k) \tag{12}$$

Implementation-wise, the first step in our algorithm is to propagate the particles from the previous time scan and then update their weights using the traditional particle filter methods. For simplicity, our evaluations below are based on the bootstrap particle filter which performs the following:

1) For $i = 1, \ldots, M$, draw samples from the motion model
$$x_{k+1}^{(i)} \sim f(x_{k+1} | x_k^{(i)}). \tag{13}$$
2) For $i = 1, \ldots, M$, compute the unnormalized weights
$$\hat{w}_{k+1}^{(i)} = w_k^{(i)} h(z_k | x_{k+1}^{(i)}) \tag{14}$$
and then normalize them as in
$$w_{k+1}^{(i)} = \frac{\hat{w}_{k+1}^{(i)}}{\sum_{j=1}^{M} \hat{w}_{k+1}^{(j)}}. \tag{15}$$
3) Under certain conditions, we perform resampling of the new particles and weights.

At this point, we have obtained samples $\{x_k^1, \cdots, x_k^M\}$ from $\bar{\omega}_{k+1}(x)$ of Eq. (8).

Next, our ASPF implementation reorders the particles according to a clustering algorithm that also helps us to calculate MMOSPA estimates of Eq. (10). A basic algorithm that does the job is given below. The variables $c_{k+1}^{(i)}$ are introduced in order to enable us to later compute p_k.

0) Set test = 0 and compute
$$\hat{x}_{k+1} = \sum_{i=1}^{M} w_{k+1}^{(i)} x_{k+1}^{(i)}. \tag{16}$$
Also set $c_{k+1}^{(i)} = 0$ for $i = 1, \ldots, M$.
1) While test = 0, do
Set test = 1
for $i = 1$ to M
if $\|\hat{x}_{k+1} - x_{k+1}^{(i)}\|^2 > \|\hat{x}_{k+1} - \chi x_{k+1}^{(i)}\|^2$
Set test = 0, $x_{k+1}^{(i)} = \chi\, x_{k+1}^{(i)}$, $c_{k+1}^{(i)} = 1 - c_{k+1}^{(i)}$.
end
end
Compute the MMOSPA estimate
$$\hat{x}_{k+1} = \sum_{i=1}^{M} w_{k+1}^{(i)} x_{k+1}^{(i)}. \tag{17}$$
end

At this point, we have obtained $\Pi_{k+1}^i \in \{I, \chi\}$ such that the samples $\tilde{x}_{k+1}^i = \Pi_{k+1}^i x_{k+1}^i$ approximate the MMOSPA posterior pdf $\tilde{\pi}_{k+1}(x)$ of Eq. (9).

Last, the probability mass that we have switched (see Eq. (11)) is given by:

$$\tilde{p}_{k+1} = \sum_{\{i : \Pi_{k+1}^i \neq I\}} w_{k+1}^i \tag{18}$$

or, implementation-wise

$$\tilde{p}_{k+1} = \sum_{i=1}^{N} w_{k+1}^{(i)} c_{k+1}^{(i)}, \tag{19}$$

and the track-to-target probability (the probability that track one is target one) is updated as in:

$$p_{k+1} = (1 - \tilde{p}_{k+1}) p_k + \tilde{p}_{k+1}(1 - p_k). \tag{20}$$

One advantage in running this algorithm is that the MMOSPA estimate $\hat{x}_{k,MMOSPA} = \hat{x}_k$ is calculated as a by-product in Eq. (17). If we are instead interested in the MMSE estimate [6] for our filter, the posterior mean is now given by

$$\hat{x}_{k,MMSE} = p_k \hat{x}_{k,MMOSPA} + (1 - p_k) \hat{x}_{k,MMOSPA}. \tag{21}$$

IV. Fast Forward Filter Unlabeled Backward Simulator (Fast FFUBSi)

The underlying weakness that we try to compensate for is that particle filters (PFs) suffer from degeneracy which leads to a self-resolving property with respect to the uncertainties in the target labeling. Thanks to recent developments [8], we can now introduce the fast FFUBSi, an efficient PF smoothing algorithm (of asymptotically linear complexity) that does not degenerate at all. In our approach, we use the forward filter to figure out where the targets are (but we don't rely on its labeling capability!) and then we use a smoothing algorithm (backwards sampling) [8] to compute/recover the labeling probabilities. It is worth underlining that this approach is dramatically different from the approach the ASPF takes and moreover, the fast FFUBSi approach is asymptotically exact, which was not the case for the approximate PF proposed in the previous section.

Smoothing for particle filters can be done through forward-backward recursions as described below [9]. The joint distribution $\pi(x_{1:T} | z_{1:T})$ at the end time T of a scenario can be decomposed as:

$$\pi(x_{1:T} | z_{1:T}) = \pi(x_T | z_{1:T}) \prod_{k=1}^{T-1} \pi(x_k | x_{k+1}, z_{1:T}) \tag{22}$$

$$= \pi(x_T | z_{1:T}) \prod_{k=1}^{T-1} \pi(x_k | x_{k+1}, z_{1:k}) \tag{23}$$

due to $\{x_k\}$ being an inhomogeneous Markov process (conditional on the measurements $z_{1:T}$).

To sample from $\pi(x_{1:T} | z_{1:T})$, one first computes the marginal distributions $\{\pi(x_k | z_{1:k})\}$ for $k = 1, \cdots, T$ in the

forward recursion and initializes the backward recursion by sampling $x_T \sim \pi(x_T|z_{1:T})$. Then, for $k = T-1, \cdots, 1$, one samples $x_k \sim \pi(x_k|x_{k+1}, z_{1:k})$ where

$$\pi(x_k|x_{k+1}, z_{1:k}) \propto f(x_{k+1}|x_k)\pi(x_k|z_{1:k}) \tag{24}$$

through Bayes rule.

The fast forward filtering backward simulator (fast FFBSi) algorithm of [8], to be described shortly in Section IV-A, is one approach to implementing the forward-backward recursion described above. In Section IV-B, we introduce modifications to the fast FFBSi that give it the additional capability of maintaining target label probabilities. We refer to the new algorithm as the fast forward filtering unlabeled backward simulator (fast FFUBSi) – we add the "U".

When keeping track of target identity probabilities, the fast FFUBSi algorithm ignores the target labels produced at the final scan by the forward filtering (which are not reliable when it comes to identifying target labels at the first scan) and relies instead on the backward trajectories generated by the backward recursion[3]. Thus, one should sample from the second term on the right hand side of Eq. (24) ignoring the labeling (more on this in Section IV-B2).

A. Fast Forward Filter Backward Simulator (Fast FFBSi)

In the following, we describe the fast FFBSi algorithm proposed in [8] which is the backbone of our fast FFUBSi. Matlab code for the implementation of the fast FFBSi algorithm can be found in [1].

We note that the original FFBSi formulation has $O(TN^2)$ complexity, where N is the number of particles. On the other hand, by taking a rejection sampling approach, the fast FFBSi algorithm was shown to have asymptotically linear complexity in the number of particles as $N \to \infty$ [1].

Input: A sequence of weighted particles systems $\{x_k^i, w_k^i\}_{i=1}^N$ describing the forward filtering distributions $\pi(x_k|z_{1:k})$, for $k = 1, \cdots, T$.

Output: A collection of backward trajectories $\{\tilde{x}_{1:T}^j\}_{j=1}^M$ describing the joint smoothing distribution $\pi(x_{1:T}|z_{1:T})$.

Algorithm:

1. Initialize the backward trajectories,
   $\{I(j)\}_{j=1}^M \sim \{w_T^i\}_{i=1}^N$,
   $\tilde{x}_T^j = x_T^{I(j)}$, $j = 1, \cdots, M$
2. for $k = T-1 : 1$
3. $\quad L = \{1, \cdots, M\}$
4. $\quad$ while $L \neq \emptyset$
5. $\quad\quad n = |L|$
6. $\quad\quad \delta = \emptyset$
7. $\quad\quad$ Sample independently[4] $\{C(q)\}_{q=1}^n \sim \{w_k^i\}_{i=1}^N$
8. $\quad\quad$ Sample independently[5] $\{U(q)\}_{q=1}^n \sim \mathcal{U}([0,1])$
9. $\quad\quad$ for $q = 1 : n$
10. $\quad\quad\quad$ if $U(q) \leq \frac{f(\tilde{x}_{k+1}^{L(q)}|x_k^{C(q)})}{\rho}$
11. $\quad\quad\quad\quad I(L(q)) = C(q)$
12. $\quad\quad\quad\quad \delta = \delta \cup \{L(q)\}$
13. $\quad\quad\quad$ end
14. $\quad\quad$ end
15. $\quad\quad L = L \backslash \delta$
16. $\quad$ end
17. $\quad$ Append the samples to the backward trajectories.
    for $j = 1, \cdots, N$
    $\quad \tilde{x}_k^j = x_k^{I(j)}$,
    $\quad \tilde{x}_{k:T}^j = \{\tilde{x}_k^j, \tilde{x}_{k+1:T}^j\}$
    end
18. end

The core of the above algorithm relies on updating L, the index list of samples at time k that still need assignment (smoothing particles) based on C, the index list of candidate samples at time k (filter particles) by testing whether the forward filter particle with (random) index $C(q)$ should be accepted as the smoothing particle with index $L(q)$ [1].

Note that there is no upper bound on the number of times the while loop on line 4 gets executed. Empirical runs show that the cardinality of L decreases rapidly in the beginning but that it takes a very long time to handle the last few particles (they get low acceptance probabilities). To circumvent this problem, the "timeout check" in [1] is added to the above algorithm. Let R_{max} be the maximum allowed number of times the while loop at line 4 can be executed. If L is not empty after R_{max} iterations, an exhaustive evaluation of the smoothing weights for the remaining elements in L is performed.

1) Upper bound on the transition density function: As mentioned previously, the fast FFBSi algorithm takes a rejection sampling approach. Hence, the assumption that the transition density function is bounded from above needs to be made:

$$f(x_{k+1}|x_k) \leq \rho \tag{25}$$

In order to maximize the acceptance probability, one wants ρ as small as possible. Let:

$$\rho = \max f(x_{k+1}|x_k) \tag{26}$$

where the maximization is over both x_{k+1} and x_k. For Gaussian models,

$$f(x_{k+1}|x_k) = \mathcal{N}(\mathrm{x_{k+1}}; \mathrm{f_k x_k}), \mathrm{R_k}) \tag{27}$$

where f is the transition density pdf and R_k is the covariance matrix of the process noise. To maximize the quantity in Eq. (27), we can simply set $x_{k+1} = f_k(x_k)$, which gives:

$$\rho = \frac{1}{\sqrt{|2\pi R_k|}} \tag{28}$$

[3]When we run the particle filters backwards in time, they do not degenerate which means that we should be able to get almost independent samples from the full trajectories.

[4]Line 7 (and line 1) of the algorithm will be modified as described in Section IV-B2 for the backward sampling to ignore the target labels produced by the forward filtering.

[5]$\mathcal{U}$ is the uniform distribution.

B. Fast FFUBSi Algorithm

The fast FFUBSi algorithm consists of running the forward filter followed by removing the target labels generated with the forward filter, backward sampling achieved through the fast FFBSi recursion described in Section IV-A and computation of target label probabilities.

1) Forward filtering: Most forward filters will have labels. However, since we are ignoring them in the backward recursion, it does not matter if they don't, that is, even if the forward filter lacks labeling, the fast FFUBSi algorithm is able to extract the labels due to its modified fast FFBSi step. In fact, we could use the same ideas to extract labels even if the forward filter is not a particle filter! For example, any MHT that performs pruning would suffer from the same self-resolving property as the particle filter. Hence, we have a good amount of freedom and flexibility when deciding on which forward filter to use for the implementation of our fast FFUBSi algorithm.

In this work, our choice for the forward filter is a standard SIR particle filter [10] with the following steps: we draw samples from the proposal density (which is the transition density function), update the particle weights based on measurement likelihoods, normalize the particle weights and resample based on these normalized particle weights.

2) Removing target labels: Let

$$\pi(x_1, x_2) = g(x_1, x_2) \tag{29}$$

be the forward density at time k for the case of two targets, which contains (not necessarily very reliable) information about the target labels at time $k = 1$. The density might be variant, invariant or a mixture of both these types[6].

Which target we decided to call x_1 and which target we decided to call x_2 in the forward filter should not influence the backward trajectories of our targets A and B. To ignore the labels produced by forward filtering, we make the forward density symmetric and replace it by:

$$\pi(x_A, x_B) = \frac{g(x_A, x_B) + g(x_B, x_A)}{2} \tag{30}$$

Notice that once we have made the forward density permutation invariant, each of the particles in the forward filter will have $n!$ copies, one for each permutation (so, two in the two target case).

3) Backward sampling: We modify the fast FFBSi algorithm that implements the backward recursion to sample instead from this symmetric forward pdf, i.e. perform rejection sampling from a set of $n! \times N$ particles (since we have $n!$ permutations of each of the N particles).

4) Computing target label probabilities: Most tracking algorithms in the literature define the target labels at time $k = 1$, with the objective to track these labeled targets over time. We instead define the target labels at time $k = T$ (at scenario end time).

[6]For definitions of permutation variant and permutation invariant pdfs, please see [11].

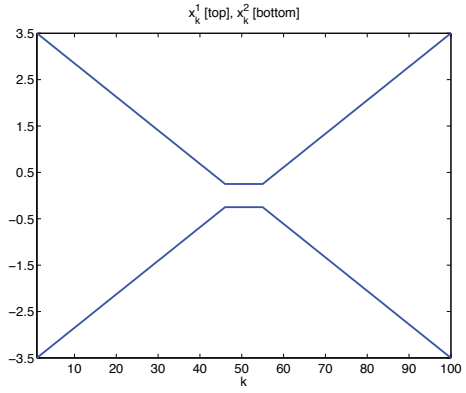


Figure 1. True target trajectories.

We offer the example in Figure 1 to clarify how the fast FFUBSi computes the probabilities of which target corresponds to which track. At time $k = 1$ we have one target at $x_1 = 3.5$ that we call target 1 and a second target at $x_2 = -3.5$ that we call target 2. We run the forward filter until the final scan $T = 100$ and can now apply the labels A and B to the two targets. Suppose that we call the target which is close $x = 3.5$ at time T target x_A and that the other target is called target x_B.

Through backward sampling, we have tracked these two targets backwards in time in order to get trajectories of how the targets that we call A and B at time $T = 100$ got to their current positions. Based on the backward trajectories, we count how many particles started assigned/close to which target and ended up assigned/close to which target, to form the probabilities of which target is which track. Then, we can calculate target label probabilities such as:
Pr (target x_1 at time $k < T$ is target x_A at time k=T) =

$$= \frac{\text{number of particles with this trajectory}}{\text{total number of particles}} \tag{31}$$

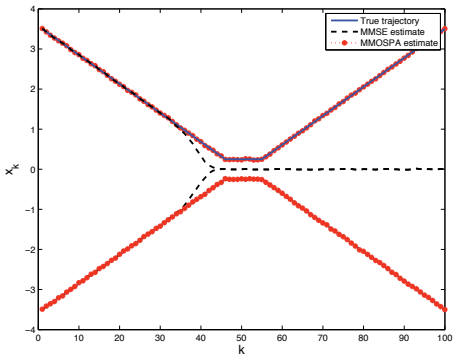
Note that our particle filters rely on the assumption that at the initial time $k = 1$ the targets are well separated. To remove the need for this assumption, we could look at the probability that $\begin{bmatrix} x_k^1 \\ x_k^2 \end{bmatrix}$ comes from the prior pdf at time $k = 1$ versus the probability that $\begin{bmatrix} x_k^2 \\ x_k^1 \end{bmatrix}$ comes from the prior.

V. Results

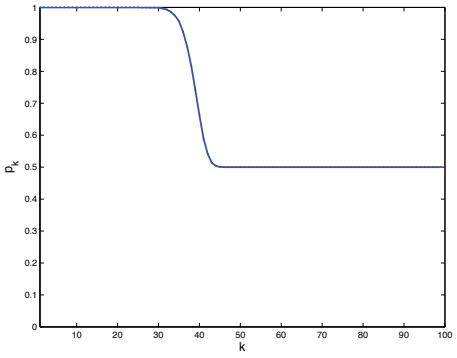
In our simulations, we consider a basic example involving two targets that come in close proximity of each other. The state vectors are scalars evolving according to a Gaussian random walk model,

$$x_k^j = x_{k-1}^j + v_{k-1}^j, \tag{32}$$

where the process noise is $v_{k-1}^j \sim \mathcal{N}(0, \sigma_v^2)$ for both targets, i.e., for $j = 1$ or 2. Both targets are detected at all times and

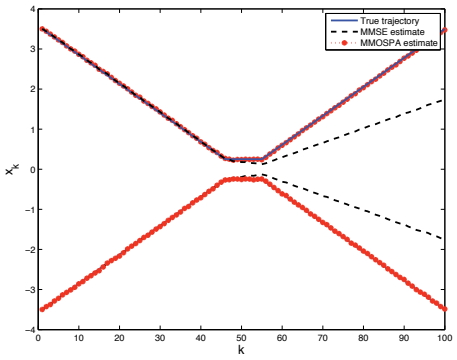


(a) True target trajectories, together with MMSE and MMOSPA estimates.

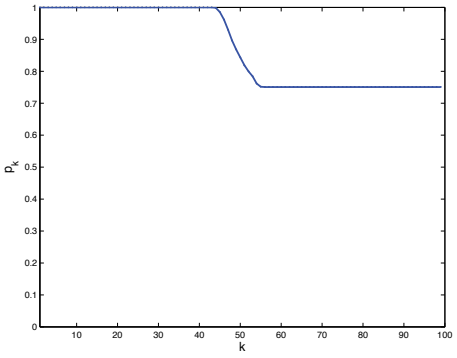


(b) Target label probabilities, p_k.

Figure 2. ASPF results for the first simulation with $\sigma_v = 1$.



(a) True target trajectories, together with MMSE and MMOSPA estimates.



(b) Target label probabilities, p_k.

Figure 3. ASPF results for the second simulation with $\sigma_v = 0.2$.

there are no false detections. Target detections are modeled as

$$z_k^j = x_k^j + u_k^j, \tag{33}$$

with measurement noise $u_k^j \sim \mathcal{N}(0, \sigma_u^2)$. However, we do not know which detection is associated to which target.

At each time instant, we have to consider two data association hypotheses:

$$H_1 : \begin{cases} \text{target 1 is associated to detection 1 and} \\ \text{target 2 is associated to detection 2} \end{cases} \tag{34}$$

$$H_2 : \begin{cases} \text{target 1 is associated to detection 2 and} \\ \text{target 2 is associated to detection 1} \end{cases} \tag{35}$$

Conditioned on a particle $x_k^{(i)}$, the measurement likelihoods for these two hypotheses are

$$L_1 = \Pr\{z_k \big| H_1, x_k^{(i)}\} \tag{36}$$

$$L_2 = \Pr\{z_k \big| H_2, x_k^{(i)}\} \tag{37}$$

or,

$$L_1 = \mathcal{N}(z_k(1); x_k(1), \sigma_v^2)\mathcal{N}(z_k(2); x_k(2), \sigma_v^2) \tag{38}$$

$$L_2 = \mathcal{N}(z_k(2); x_k(1), \sigma_v^2)\mathcal{N}(z_k(1); x_k(2), \sigma_v^2). \tag{39}$$

Given the above likelihoods, we compute the measurement likelihood given the particle as

$$h(z_k|x_k^{(i)}) = \sum_{j=1}^{j=2} L_j \Pr\{H_j \big| x_k^{(i)}\} = \frac{1}{2}\sum_{j=1}^{j=2} L_j, \tag{40}$$

We use Eq. (40) to perform the weight update steps in both ASPF and the fast FFUBSi.

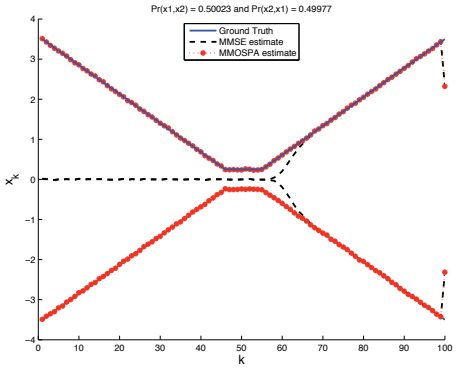
As metrics of performance, we chose to look at MMSE [6] and MMOSPA [7] estimates together with target label probabilities. The following figures show the averaging of 100 Monte Carlo runs.

A. ASPF Results

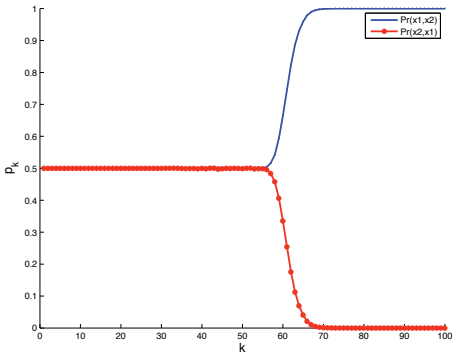
In our simulation for the ASPF, the two targets are well separated except for a few time steps in the middle. More specifically, the target trajectories start at a distance of 7 (arbitrary units), spend 10 time steps at a distance of 0.5 during the middle part of the sequence, then separate again, as seen in Fig. 1.

We assume that the states of target x_1 and x_2 are well known at time $k = 1$ and use $\sigma_u = 0.1$ and $M = 2000$ particles.

In the first simulation, we use $\sigma_v = 1$ and we can see that the p_k converges to 0.5 over time, as in Fig. 2. We also notice that the MMOSPA estimates are close to the true states whereas the MMSE estimates coalesce somewhat.

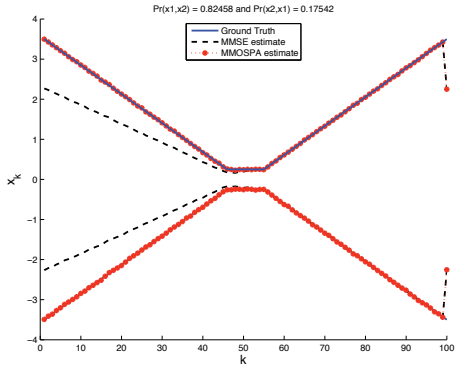


(a) True target trajectories, together with MMSE and MMOSPA estimates.

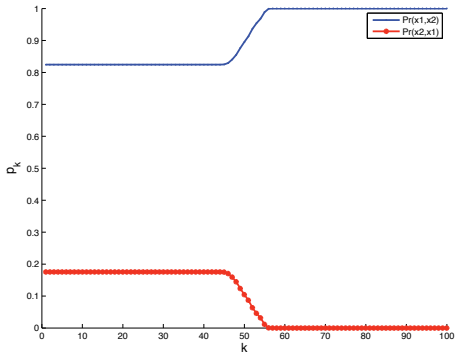


(b) Target label probabilities, p_k.

Figure 4. Fast FFUBSi results for the first simulation with $\sigma_v = 1$.



(a) True target trajectories, together with MMSE and MMOSPA estimates.



(b) Target label probabilities, p_k.

Figure 5. Fast FFUBSi results for the second simulation with $\sigma_v = 0.2$.

In the second simulation, we decrease the scenario's difficulty by using $\sigma_v = 0.2$, which means that the predicted densities overlap less. Hence, the posterior density of the targets should not be as permutation symmetric as in the first simulation.

We notice in Fig. 3 that with $M = 2000$ particles the tracks do not coalesce and p_k remains large throughout the whole scenario. Consequently, the MMOSPA estimates perform very well and the MMSE estimates perform satisfactorily for this filter on this scenario.

B. Fast FFUBSi Results

In the simulations for the fast FFUBSi, we considered the same scenario (i.e. the same data) as in the simulations for the ASPF although a direct comparison between the two approaches is not possible[7]. We use the same number of smoothing particles, i.e., 2000 as the number of particles in the forward filter. Please remember that the MMOSPA estimates and MMSE estimates shown in Fig. 4 and Fig. 5 are now computed based on the backward sampling.

[7]The ASPF takes the traditional approach and labels targets at time $k = 1$ and then proceeds (forward) to calculate $Pr(l_{k>1} = l_{k=1})$, where l_k are the target labels at time k. On the other hand, the fast FFUBSi labels the targets at time $k = T$ and then proceeds (backward) to calculate $Pr(l_{k<T} = l_{k=T})$.

In the first simulation for the fast FFUBSi, we use the same $\sigma_v = 1$ value as in the first simulation performed for the ASPF and we can see that the p_k converges to 0.5 over time, see Fig. 4.a and Fig. 4.b. We also notice that the MMOSPA estimates are close to the true states throughout the length of the scenario whereas the MMSE estimates coalesce after backward sampling from the multimodal posterior pdf characteristic of the period in which targets are in close proximity.

In the second simulation, we use $\sigma_v = 0.2$. This is an easier scenario, reflected in the p_k values not reaching the permutation variant point of 0.5. Figure 5 shows excellent MMOSPA estimates at all time steps while the backward MMSE estimates display a coalescence tendency, albeit less strong than for the first simulation.

VI. Conclusions

In this work, we considered the problem of extracting information about target labels in the case of two targets that come in close proximity of each other and measurements that do not provide information about target identities. In response, we have developed two novel particle filters of linear complexity in the number of particles that take clearly different approaches to remembering uncertainties in track labeling. On one hand, the Approximate Set Particle Filter

(ASPF) recursively updates the global probabilities of different track-to-target associations while at the same time computing MMOSPA estimates. On the other hand, the fast Forward Filter Unlabeled Backward Simulator (fast FFUBSi) calculates target label probabilities based on backward trajectories output by a smoothing technique.

Simulations were performed for a simple scenario of two targets that start separated, move close to each other and separate again. The effect of process noise on the target label probabilities, p_k was investigated and their behavior for both filters matched theoretical expectations, i.e. increasing process noise leads to increasing uncertainty in target identities as reflected in the target label probabilities moving towards 0.5. Additionally, MMOSPA estimates were seen to be preferable to MMSE estimates throughout.

The next goal for this work is to extend the ASPF and fast FFUBSi to cover the case of three or more targets. Generalization to a variable and unknown number of targets should follow.

ACKNOWLEDGMENT

R. Georgescu and P. Willett were supported by the Office of Naval Research under contract N00014-10-10412.

REFERENCES

[1] F. Lindsten, "Rao-blackwellised particle methods for inference and identification," Master's thesis, Linkoping University, Linkoping, Sweden, 2011.

[2] Y. Boers and H. Driessen, "The mixed labeling problem in multi target particle filtering," in *Proc. of 10th International Conf. on Information Fusion*, Quebec City, Canada, 2007.

[3] H. A. P. Blom and E. A. Bloem, "Decomposed particle filtering and track swap estimation in tracking two closely spaced targets," in *Proc. of 14th International Conf. on Information Fusion*, Chicago, IL, 2011.

[4] A. F. Garcia-Fernandez, M. R. Morelande, and J. Grajal, "Particle filter for extracting target label information when targets move in close proximity," in *Proc. of 14th International Conf. on Information Fusion*, Chicago, IL, 2011.

[5] M. Skolnik, *Introduction to Radar Systems*. New York, NY: McGraw-Hill, 2002.

[6] Y. Bar-Shalom and X. R. Li, *Multitarget-Multisensor Tracking: Principles and Techniques*. Storrs, CT: YBS Publishing, 1995, pp. 402–437.

[7] M. Guerriero, L. Svensson, D. Svensson, and P. Willett, "Shooting two birds with two bullets: how to find Minimum Mean OSPA estimates," in *Proc. of 13th International Conf. on Information Fusion*, Edinburgh, Scotland, 2010.

[8] R. Douc, A. Garivier, E. Moulines, and J. Olsson, "Sequential Monte Carlo smoothing for general state space hidden Markov models," *Submitted to Annals of Applied Probability*, 2010.

[9] A. Doucet and A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," in *Oxford Handbook of Nonlinear Filtering*, D. Crisan and B. Rozovsky, Eds. Oxford, UK: Oxford University Press, 2009.

[10] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for on-line non-linear/non-Gaussian Bayesian tracking," *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–189, 2002.

[11] D. Crouse, P. Willett, and Y. Bar-Shalom, "Generalizations of Blom and Bloem's PDF decomposition for permutation-invariant estimation," in *Proc. of Intl. Conf. on Acoustics, Speech and Signal Process.*, Prague, Czech Republic, May 2011.