Sequential Monte-Carlo methods for Multi-target Filtering

Gwendoline DE BIE Johann FAOUZI Hicham JANATI

***Abstract*- In this paper, we focus on the multi-target filtering problem through a study of the sequential Monte-Carlo implementation of the Probability Hypothesis Density (PHD) filter proposed in [1]. After reviewing its main underlying theoretical concepts, we present our own implementation, detail our results and reflect on key implementation choices.**

Introduction

In the most general sense, multi-target tracking aims at estimating an unknown, time-varying number of targets and their states from a discrete sequence of noisy observations. This problem is in essence challenging because there is uncertainty over which target generated which observation, and over whether these measurements are false alarms. The PHD filter was proposed as a solution and is based on Mahler's finite set statistics (FISST) theory. Whereas the optimal multi-target Bayes filter propagates the multi-target posterior density, whose computational cost can be really prohibitive, the PHD filter, which propagates the first statistical moment of the multiple target posterior distribution called PHD, appears as an efficient sub-optimal alternative. We have focused on studying a sequential Monte-Carlo implementation of the PHD filter, namely [1], which provides with a practical solution to the PHD recursion. We present here our own implementation choices and programs and results after stating some core theoretical elements.

1 Core theoretical elements of the PHD filter

1.1 The use of random finite sets

The framework adopted in [1] represents the multi-target states and observations as finite sets X_k and Z_k, which respectively represent targets' positions and measurements. Indeed, this representation allows all possible occurences of the multi-target states and comes with a consistent and meaningful notion of error, which would not be the case in a vectorial representation in an Euclidean space for instance. Moreover, in a Bayesian paradigm, states and observations are treated as realizations of random variables, which legitimates the use of *random finite sets.* As underlined in [3], modeling individual motion of a target in a multi-target environment involves constructing multi-target transition and density from individual density on a single-target state space and individual likelihood on a single-target observation space, which is challenging, whereas resorting to FISST allows direct Bayesian inferencing through the use of belief mass functions. In [1] however, links are made between FISST and conventional probabilities, which enables us to use its underlying concepts while resorting to conventional probabilities in the following.

1.2 Problem formulation

As hinted at earlier on, let us denote by X_k and Z_k the finite sets representing the multi-target states and observations respectively, Ξ_k and Σ_k being the related random finite sets (RFS). With obvious notations, the number of targets at time k stems from the ones that have survived from time $k-1$ and the newborn ones as we have neglected spawning, which is modeled by $\Xi_k = S_k(X_{k-1}) \cup N_k(X_{k-1})$. Similarly, observations at time k stem from measurements generated by X_k or false alarms, which is written as $\Sigma_k = O_k(X_k) \cup F_k(X_k)$. The multi-target filtering problem aims at estimating the multi-target state X_k given all the observations $Z_1, \dots, Z_k$, by propagating the first moment of the RFS Ξ also called *intensity function* D_k associated with the multi-target posterior in time, through the recursion :

$$D_{k|k-1}(x) = \gamma_k(x) + \int \phi_{k|k-1}(x,\xi) D_{k-1}(\xi)\lambda(\mathrm{d}\xi) \qquad (1)$$

$$D_k(x) = (1 - p_D(x)) D_{k|k-1}(x) + \sum_{z \in Z_k} \frac{\psi_{k,z}(x) D_{k|k-1}(x)}{\kappa_k(z) + \int \psi_{k,z}(\xi) D_{k|k-1}(\xi)\lambda(\mathrm{d}\xi)} \qquad (2)$$

where equation (1) is referred to as the *prediction step*, characterized by γ_k, the intensity function parameterizing the birth of targets, and by the transition density $\phi_{k|k-1}$, which is the product of the single-target transition density of the model $f_{k|k-1}$ and the probability $p_{S,k}$ of target survival, as we have neglected spawning ; where equation (2) is called the *update step*, characterized by the probability of detection p_D, the intensity κ_k related to the measurements, and where $\psi_{k,z}$ is the product of the probability of detection p_D and the likelihood g_k of the individual targets in the model.

This recursion is based on the assumptions that the targets evolve independently of each other, that the detection and measurement of a target is independent from that of other targets, and that the predicted multi-target density is Poisson[1].

2 A sequential Monte-Carlo implementation of the PHD filter

2.1 Particle approximations

Since sampling directly from the multi-target posterior would be impossible, in the sequential Monte-Carlo rationale, we resort to *particle paths*, which provides us with a *particle representation* of the intensity function, of the form $\hat{D}_{k-1}(x_{k-1}) = \sum_{i=1}^{L_{k-1}} w_{k-1}^{(i)} \delta_{x_{k-1}^{(i)}}(x_{k-1})$, where the particles are drawn from convenient importance sampling densities. Particles are carrying out targets estimation, therefore their number also varies as the sum of the remaining and newborn ones, namely $L_k = L_{k-1} + J_k$ with obvious notations.

Using this particle representation and denoting p_k and q_k the importance sampling densities related to $\phi_{k|k-1}$ and γ_k respectively, the Monte-Carlo approximation of the *prediction step* can be written as

$$\hat{D}_{k-1}(x_{k-1}) = \sum_{i=1}^{L_{k-1}} w_{k|k-1}^{(i)} \delta_{x_{k-1}^{(i)}}(x_{k-1}) \tag{3}$$

where the remaining particles are sampled from q_k and the newborn ones from p_k, and where the weights are defined as

$$w_{k|k-1}^{(i)} = \begin{cases} \dfrac{\phi_{k|k-1}(x_k^{(i)}, x_{k-1}^{(i)}) w_{k-1}^{(i)}}{q_k(x_k^{(i)}|x_{k-1}^{(i)}, Z_k)}, & \text{for the remaining particles at this step} \\ \dfrac{\gamma_k(x_k^{(i)})}{J_k p_k(x_k^{(i)}|Z_k)}, & \text{for newborn ones.} \end{cases} \tag{4}$$

which stems from equation (1) above.

Similarly, the *update step* turns the particle representation of the targets from $\{w_{k|k-1}^{(i)}, x_k^{(i)}\}_{i=1}^{L_{k-1}+J_k}$ into $\{w_k^{(i)}, x_k^{(i)}\}_{i=1}^{L_{k-1}+J_k}$ by modifying the weights according to

$$w_k^{(i)} = (1 - p_D(x_k^{(i)})) w_{k|k-1}^{(i)} + \sum_{z \in Z_k} \frac{\psi_{k,z}(x_k^{(i)}) w_{k|k-1}^{(i)}}{\kappa_k(z) + C_k(z)} \tag{5}$$

which directly stems from equation (2), where $C_k(z) = \sum_{j=1}^{L_{k-1}+J_k} \psi_{k,z}(x_k^{(j)}) w_{k|k-1}^{(j)}$.

This way, the number of targets N_k which is equal to the total mass $\int D_k(\xi)\lambda(\mathrm{d}\xi)$ can be approximated through $\hat{N}_k = \sum_{j=1}^{L_{k-1}+J_k} w_k^{(j)}$. However, since we sample from a very high dimensional state-space, the weights become quickly highly degenerate. We tackle this issue through *resampling*, which implies some additional variance in the short run, but provides with stability and avoids accumulation of errors over time[2]. The outline of the algorithm, which is detailed in Appendix, consists in the steps presented above in that order. In addition, a clustering step is needed to extract the information on targets states based on the weighted particles.

2.2 Model description and implementation choices

Our *Python* implementation of the PHD filter involves *classes*, which are a convenient tool to encapsulate all the parameters and related functions associated to the model. As hinted at earlier on, we have assumed for the sake of simplicity that no spawning is involved, and that all captors are ideal, hence a probability of detection $p_D = 1$. We have used a similar model to the one chosen in [1] : targets are observed over the region $[-100; 100] \times [-100; 100]$ and move according to the following Gaussian dynamics :

$$x_k = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} x_{k-1} + \begin{bmatrix} \frac{T^2}{2} & 0 \\ T & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \end{bmatrix} \begin{bmatrix} v_{1,k} \\ v_{2,k} \end{bmatrix}$$

where $T = 1$, x_k is a four-sized vector containing position and velocity coordinates, and where $(v_{1,k}, v_{2,k})$ is a zero-mean mutually independent Gaussian noise with respective standard deviations $(2, 2)$. The transition density $f_{k|k-1}$ is defined accordingly. In the model, each target has a constant $p_{S,k} = 0.99$ probability of survival. The birth of new targets is parameterized through a Poisson point process with intensity function $\gamma_k = 0.05\mathcal{N}(a, Q)$, where :

$$a = \begin{bmatrix} 0 \\ 3 \\ 0 \\ -3 \end{bmatrix} \text{ and } Q = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Measurements naturally depend on positions rather than velocity, as done in :

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

where $(w_{1,k}, w_{2,k})$ is a zero-mean mutually independent Gaussian noise with respective 2.5 standard deviations. The likelihood g_k is defined accordingly. Clutter is introduced in the model through parameter r, which basically

[1]The latter assumption yields a closed form expression for the update step of the PHD filter.
[2]"Sacrificing the past to save the future".

stands for κ_k and accounts for the average number of false alarms. The number of particles naturally varies throughout the process. However, a constant number of ρ particles are used for each expected target and J particles are generated at each iteration to account for the possible birth of new targets. The latter does not weigh much on the algorithm since those particles are deleted at the resampling step if no particle was born at the beginning of the step, which provides with a *soft* limit on their number.

At the end of each iteration, target states are extracted through a k-means clustering step on the weighted particles. This step basically aims at determining which particles belong to which target, according to their spatial position at time k, and setting the target positions to be the centers of the clusters. A threshold has been added as a parameter, which can take the form of a percentile or an arbitrary proportion, in order to select the most relevant ones before computing the clustering.

Finally, the notion of error measurement, that motivated the use of *random finite sets* in the first place, is crucial and challenging since filtering involves estimating both the number of constituents and their values. As underlined in [3], this metric should apply on finite sets, capture both cardinality and state errors meaningfully and be computed rather easily. In that spirit, the optimal transport framework provides with the Wasserstein distance, which in our context is defined as follows :

$$d_p(X,Y) = \min_C \left(\sum_{i=1}^{|X|} \sum_{i=1}^{|Y|} C_{i,j} \|y_j - x_i\|^p \right)^{1/p}$$

which we compute between the true states X and the estimated ones $\hat{X}$. It computes the L^p error when moving the distribution of X onto the one of $\hat{X}$ with minimal cost, and is therefore also known as the *earth mover's distance.*

As a comparison, we also compute the *Optimal SubPattern Assignment* (OSPA) metric, defined as follows, where $|\cdot|$ denotes set cardinality, $|X| < |Y|$ and $d^{(c)}(x,y) = \min(c, d(x,y))$[3] :

$$d_p^{(c)}(X,Y) = \left(\frac{1}{|Y|} \left(\min_{\pi \in \Pi_{|Y|}} \sum_{i=1}^{|X|} d_p^{(c)}(x_i, y_{\pi(i)}) + c^p(|Y| - |X|) \right) \right)^{1/p}$$

which, as detailed in [3], solves some drawbacks of the Wasserstein distance, including a dependence on the geometry of the points and some inconsistencies in the case of cardinality differences. Even though we may be reluctant to add an extra hyperparameter to the model on first thought, it proves efficient in the end as the second term of the sum enables us to detect cardinality differences between estimated and actual targets very easily with large values[4] of c, which can be of great use in terms of efficiency assessment.

Throughout this process, a key attention point is the choice of importance sampling densities. Natural choices of q_k and p_k lead to $q_k \propto \phi_{k|k-1}$, which is equivalent to $q_k \propto f_{k|k-1}$ since spawning has been left out, and $p_k \propto \gamma_k$. These choices contain all the information that is available in the model regarding the birth process and the transition density respectively, in addition to which they fall under Geweke's sufficient condition for finite variance, and lead to interesting simplifications in the particles' weight update formulas, see 4. Despite these very appealing properties, the valuable information contained in the observations remains unused at the prediction step with these choices of proposals. Moreover, poor performance may occur due to their lack of robustness in presence of outliers. In order to reduce the mismatch, particles could be proposed under the following distribution, inspired by [4]:

$$q_k(x_k|x_{k-1}, z_k) = \frac{f_{k|k-1}(x_k|x_{k-1}) g_k(z_k|x_k)}{\int f_{k|k-1}(x|x_{k-1}) g_k(z_k|x) \mathrm{d}x} \quad (6)$$

$$p_k(x_k|z_k) = \frac{\gamma_k(x_k) g_k(z_k|x_k)}{\int \gamma_k(x) g_k(z_k|x) \mathrm{d}x} \quad (7)$$

Even though (6) and (7) are intractable in most cases, it is not as bad in the Gaussian case[5], where the product of two normal densities is proportional to one.

Another strategy developed in [5] would be to compute q_k and p_k minimizing the variance of the particles' weights, preventing degeneracy at the same time, which we have not pursued on the grounds of complexity. Finally, several proposals have been made in the unscented sequential Monte-Carlo setting, that includes the covariance of transition model in sample representations. For instance, [6] goes a bit further by presenting an enhanced approximation of the importance sampling densities through the use of an unscented information filter with a gating technique to ensure robust estimations.

3 Analysis of results

3.1 Simulation

As detailed in the previous section, we have reproduced the first simulation example of [1] with a slightly different set of parameters, which ensures the use of wider paths within the observation region. We also set the clutter rate to $r = 10$. Figure 1 displays the true positions (circles), the observations (crosses) and the false alarms (dots). We run our bootstrap PHD filter with $\rho = 200$ and $J = 50$. We can see in Figure 2 that the different paths are retrieved even with a low number of particles per target. Figure 3 shows however some irregularities when estimating the number of targets. The wrong predictions are only due to

[3]Where we have chosen d to be a p-norm, $p = 2$ in all numerical applications presented here.

[4]We have opted for $c = 10$ in all numerical applications presented here.

[5]Which we focus on in the implementation, as developed above.

the presence of false alarms: when setting r=0, the PHD filter never fails to predict the correct number of targets.

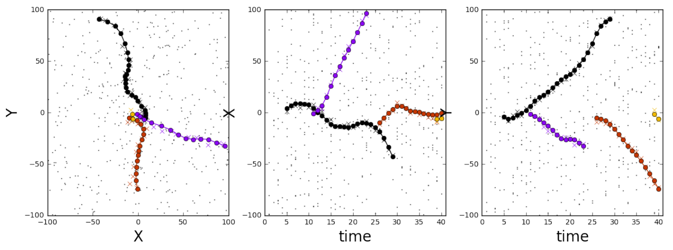

Figure 1: *True and observed positions with r = 10*

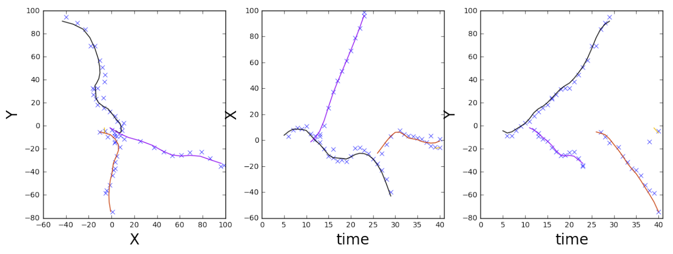

Figure 2: *Predicted positions (crosses) vs true positions (circles)*

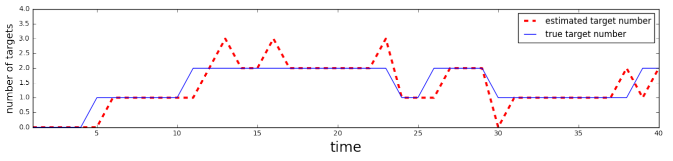

Figure 3: *Estimated number of targets vs true number of targets*

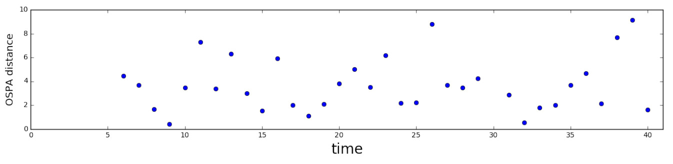

Figure 4: *OSPA metric*

The OSPA metric shows very high values (that mostly happen around the wrong estimations of the number of targets). These rather poor results can be explained by the high variance of the PHD filter when a small number of particles is used, and by the metric itself, which is designed to take large values when errors in target numbering occur. Moreover, a substantial part of the points are below the root-mean-squared measurement noise, approximately equal to 3.5 in our model. The effect of ρ and J is discussed in the following paragraph.

3.2 Effect of the number of particles

Proposition 3 of [1] guarantees that when $L \to \infty$, the mean of the particle filter converges to the PHD density. Here, we launch 50 predictions for three different sets of prediction parameters. We compute the mean of the OSPA curves and their standard deviation. Figure 5 shows that when the number of particles is large, the OSPA metric is *lower* and has smaller variance.

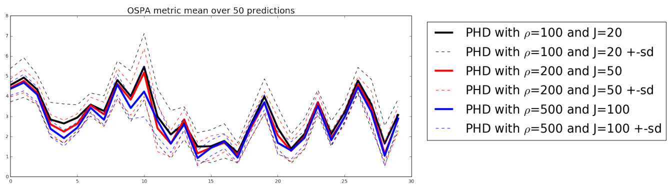

Figure 5: *OSPA metric mean over 50 predictions*

3.3 Prediction method

After resampling the particles and performing clustering, the state estimates can be computed as:

- cluster centroids
- maximum weight particles

Our implementation of the PHD filter gives the user the possibility to choose either of the two. As in the previous comparison, we compute the OSPA metric and the interval curves [mean - sd, mean + sd]. Figure 6 shows that the centroids method is slightly better than the maximum weight prediction.

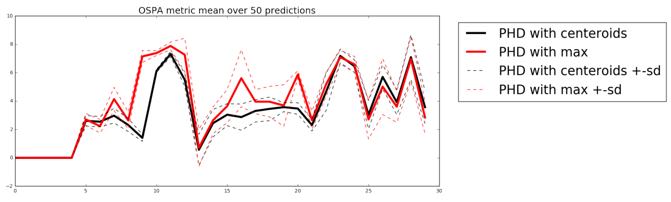

Figure 6: *OSPA metric mean over 50 predictions*

All theses simulations can be found in the *Jupyter* notebooks enclosed with this report.

4 Extensions

At each time step, the presence of false alarms makes the observations very noisy. Thus, using raw observations in the proposal q_k is questionable. On the contrary, not using them does not seem ideal either since it leaves some available and meaningful information unused. To solve this issue, we could think of computing a two-stage PHD filter. At the first stage, using $q_k = f_{k|k-1}$, the number of targets and their positions is predicted. Then, one associates the predicted to the observed positions. If an observed position is left on its own, it is considered to be a false alarm, thus enabling their detection. A two-stage PHD filter would consist in removing the predicted false alarms at the end of the first stage. Then, at the second stage, if one restricts the observations to the ones that have not been categorized as false alarms, the set of observations is much less noisy. One can finally compute the PHD filter with importance sampling density q_k, taking into account the observations this time.

Conclusion

This *sequential* implementation of the PHD filter provides very appealing properties of both theoretical and computational natures. Its efficiency is however highly dependent on many rather arbitrary choices set by the user, whether it be of *theoretical* nature (notion of error measurement, choice of importance sampling densities or clustering method) or *practical* nature (choice of the main hyperparameters). Our *bootstrap* implementation of the PHD filter, notably characterized by the *Optimal Sub-Pattern Assignment* (OSPA) metric and state estimated estimated through k-means cluster centroids, succeeds in retrieving the targets' paths and detecting false alarms, which is quite impressive considering the generality of the framework used, as opposed to the specificity of [2].

Appendix

PHD filter algorithm

```
Algorithm 1: PHD filter at each time step $k$
  Data: $\tilde{x}_{k-1}$, $\tilde{w}_{k-1}$, $Z_k$
  Result: $\hat{X}_{k-1}$, $\tilde{x}_k$, $\tilde{w}_k$
  Step 1: Prediction
    • for $i = 1, \dots, L_{k-1}$ do
        $\tilde{x}_k^{(i)} \sim q_k(\cdot|\tilde{x}_k^{(i)}, Z_k)$
        $\tilde{w}_{k|k-1}^{(i)} = \frac{\phi_{k|k-1}(\tilde{x}_k^{(i)}, \tilde{x}_k^{(i)})}{q_k(\tilde{x}_k^{(i)}|\tilde{x}_k^{(i)}, Z_k)}$
    • for $i = L_{k-1} + 1, \dots, L_{k-1} + J_k$ do
        $\tilde{x}_k^{(i)} \sim p_k(\cdot|Z_k)$
        $\tilde{w}_{k|k-1}^{(i)} = \frac{1}{J_k} \frac{\gamma_k(\tilde{x}_k^{(i)})}{p_k(\tilde{x}_k^{(i)}|Z_k)}$
  Step 2: Update
    • for each $z \in Z_k$ do
        $C_k(z) = \sum_{i=1}^{L_{k-1}+J_k} \psi_{k,z}(\tilde{x}_k^{(i)}) \tilde{w}_{k|k-1}^{(i)}$
    • for $i = 1, \dots, L_{k-1} + J_k$ do
        $\tilde{w}_k^{(i)} = \left[ \sum_{z \in Z_k} \frac{\psi_{k,z}(\tilde{x}_k^{(i)})}{\kappa_k(z) + C_k(z)} \right] \tilde{w}_{k|k-1}^{(i)}$
  Step 3: Resampling
    • Compute the total mass $\hat{N}_k = \sum_{i=1}^{L_{k-1}+J_k} \tilde{w}_k^{(i)}$
    • Resample $\left\{ \frac{\tilde{w}_k^{(i)}}{\hat{N}_k}, \tilde{x}_k^{(i)} \right\}_{i=1}^{L_{k-1}+J_k}$ to get $\left\{ \frac{\tilde{w}_k^{(i)}}{\hat{N}_k}, \tilde{x}_k^{(i)} \right\}_{i=1}^{L_k}$
    • Multiply the weights by $\hat{N}_k$ to get $\left\{ \tilde{w}_k^{(i)}, \tilde{x}_k^{(i)} \right\}_{i=1}^{L_k}$
  Step 4: Clustering
  Round the total mass to estimate the number of
  targets and use a clustering algorithm with
  n_cluster = round($\hat{N}_k$). Compute $\hat{X}_k$ as a function
  of each cluster.
```

References

[1] B.-N. Vo, S. Singh, A. Doucet, *Sequential Monte-Carlo methods for Multi-target Filtering with Random Finite Sets*, 2005

[2] D. Clark, K. Panta, B.-N. Vo, *The GM-PHD Filter Multiple Target Tracker*, 2006

[3] B.-T. Vo, *Random Finite Sets in Multi-Object Filtering*, 2008

[4] O. Cappé, S. J. Godsill, E. Moulines, *An overview of existing methods and recent advances in sequential Monte-Carlo*, 2007

[5] N. Whiteley, S. Singh, S. Godsill, *Auxiliary Particle Implementation of the Probability Hypothesis Density Filter*, 2007

[6] J.-H. Yoon, D.-Y. Kim, K.-J. Yoon, *Efficient importance sampling function design for sequential Monte Carlo PHD filter*, 2012