Fast Particle Flow Particle Filters via Clustering

Yunpeng Li and Mark Coates
Department of Electrical and Computer Engineering, McGill University, Montréal, Québec, Canada
Email: yunpeng.li@mail.mcgill.ca;mark.coates@mcgill.ca

***Abstract*—Particle flow filters, introduced in a series of papers by Daum and Huang, are an attractive alternative to particle filters for filtering tasks in high-dimensional spaces or with very informative measurements. Many variants of particle flow filters have been developed, but all require approximations in multiple stages of the implementation, which leads to particles deviating from the true posterior distribution. To preserve the statistical consistency of the filtering algorithm, some recent papers embed the particle flow techniques within a particle filter, using them to generate a proposal distribution. In recent work, we developed such a particle flow particle filter, modifying the flow mechanism to ensure that the implemented, approximate flow was an invertible mapping. This property allows efficient computation of the importance weights. In this paper, we strive to reduce the computational overhead of the particle flow particle filter by incorporating clustering of the particles. Results from a multi-target acoustic tracking simulation demonstrate that we can significantly reduce the computational cost of particle flow particle filters with a relative small sacrifice in tracking accuracy.**

I. Introduction

Particle filters have been standard tools in filtering with nonlinear and non-Gaussian models. However, the weight degeneracy issue has plagued their application when the state dimension is high or when measurements are highly informative [1]. In a series of recent papers, Daum and Huang introduced particle flow filters, which migrate particles from the prior distribution to the posterior by constructing "flows" for each particle [2], [3]. There is no importance sampling step so the weight degeneracy issue is avoided.

However, computationally tractable solutions only exist for a few of the proposed particle flow filters. The *exact Daum and Huang* (EDH) filter, proposed in [3], identifies update equations for the case when the prior and posterior are both Gaussian and the measurement model is linear. These equations can be applied to nonlinear models via a linearizing approximation. As described in [3], the EDH filter computes the particle flow at the mean of the evolving particle cloud, and applies it to all particles. A minor variation was explored in [4], with individual particle flow updates evaluated for each particle. This *localized exact Daum and Huang* (LEDH) filter is much more computationally demanding but can provide more accurate state estimates.

All particle flow filters require approximations in various stages of implementation. Particles thus are in general not exactly distributed according to the posterior distribution after completion of the flow, and the discrepancy with the true posterior is not well understood. Thus, the omission of the importance sampling comes at a cost of sacrificing the statistical consistency of the filter. An alternative approach is to incorporate the particle flow method within the particle filtering framework. In [5], the drift homotopy technique is incorporated within a Markov chain Monte Carlo (MCMC) step to sample from the true posterior. Other algorithms have used the particle flow approach to construct a proposal distribution; the importance sampling (or weight update) step is still applied and corrects for the discrepancy between the posterior and the proposal distribution. The Gaussian particle flow importance sampling algorithm (GPFIS) [6] constructs an approximate Gaussian flow to sample from nonlinear Gaussian models; the weight update is performed at each intermediate time step of the flow. An approach based on optimal transport concepts in [7] identifies a coupling between the prior and the posterior and combines particle transport with importance sampling. In [8], the stationary solution of the Fokker-Planck equation is used to derive a stochastic particle flow. All of these approaches mentioned above are elegant but are orders-of-magnitude more computationally complex than the original particle flow algorithms, e.g., the EDH and LEDH.

Recently, we proposed particle flow particle filters (PF-PF) that incorporated a deterministic particle flow possessing the invertible mapping property [9]. The invertible mapping property allows us to perform a very efficient weight update. The PF-PF based on LEDH has almost the same computational overhead as LEDH (the weight update is much less demanding than the flow evaluation). We provided a numerical simulation example illustrating that the PF-PF based on LEDH can provide more accurate estimates than the LEDH [9], suggesting that incorporating the importance sampling correction is beneficial.

In this paper, we propose a fast particle flow particle filter by first identifying clusters of particles and assigning each particle to a cluster. The expensive calculation of the flow is performed only at the cluster medoids. As the number of clusters is smaller than the number of particles, we expect reduction of the computational cost. We design distance metrics for clustering within the particle flow particle filter framework, and examine the trade-off between the computational cost and the tracking accuracy in a challenging multi-target acoustic tracking simulation.

The rest of the paper is organized as follows. Section II states the filtering task. Section III provides a brief review of the particle flow technique that is incorporated in the proposed filter. We introduce the proposed particle cluster flow particle filters in Section IV. Section V introduces the simulation setup and presents results. The conclusion is provided in Section VI.

II. Problem Statement

We consider the nonlinear filtering task with the following models:

$$x_n = g(x_{n-1}, v_n) \tag{1}$$

$$z_n = h(x_n, w_n)\,. \tag{2}$$

where $g : \mathbb{R}^D \to \mathbb{R}^D$ is the dynamic model involving the unobserved state $x_n \in \mathbb{R}^D$ and the process noise $v_n \in \mathbb{R}^D$. The nonlinear measurement model $h : \mathbb{R}^D \to \mathbb{R}^S$ describes the relation between the state x_n and the measurement $z_n \in \mathbb{R}^S$. $w_n \in \mathbb{R}^S$ is the measurement noise. Our goal is to track the marginal posterior distribution $p(x_n|z_1, \ldots, z_n)$.

III. Background

The particle flow process can be modeled as a background stochastic process η_λ for pseudo-time $\lambda \in [0, 1]$, between the time steps $n-1$ and n. The distribution of η_0 is the prior of x_n and the distribution of η_1 is the posterior of x_n. The flow of the i-th particle η^i_λ is considered as the i-th realization of this stochastic process.

When both the prior and the posterior are Gaussian and the measurement function is linear, the EDH filter of [3] gives an exact expression of the drift term in a deterministic flow:

$$\frac{d\eta}{d\lambda} = \zeta(\eta, \lambda) = A(\lambda)\eta + b(\lambda) \tag{3}$$

where

$$A(\lambda) = -\frac{1}{2}PH^T(\lambda HPH^T + R)^{-1}H, \tag{4}$$

$$b(\lambda) = (I + 2\lambda A)[(I + \lambda A)PH^T R^{-1} z_k + A\bar{\eta}_0]. \tag{5}$$

P is the covariance matrix of the prediction error for the prior distribution, and H is the measurement matrix. For nonlinear models, H is an approximation of $h(\cdot, \cdot)$ derived through the linearization of the measurement model, i.e. $H = \dfrac{\partial h(\eta, 0)}{\partial \eta}$. R is the covariance matrix of the measurement error, and $\bar{\eta}_0$ is the mean of the prior distribution. The flow is performed by iterating through N_λ discrete time steps with step sizes $\Delta\lambda(1), \ldots, \Delta\lambda(N_\lambda)$.

The LEDH filter calculates $A^i(\lambda)$ and $b^i(\lambda)$ of the drift term $\zeta(\eta^i_\lambda, \lambda) = A^i(\lambda)\eta^i_\lambda + b^i(\lambda)$ for each individual particle η^i_λ, where

$$A^i(\lambda) = -\frac{1}{2}PH^i(\lambda)^T(\lambda H^i(\lambda)PH^i(\lambda)^T + R)^{-1}H^i(\lambda), \tag{6}$$

$$b^i(\lambda) = (I + 2\lambda A^i(\lambda))[(I + \lambda A^i(\lambda))PH^i(\lambda)^T R^{-1}(z - e^i(\lambda)) + A^i(\lambda)\bar{\eta}_0]. \tag{7}$$

Here $e^i(\lambda) = h(\eta^i_\lambda, 0) - H^i(\lambda)\eta^i_\lambda$ where $H^i(\lambda) = \left.\dfrac{\partial h(\eta, 0)}{\partial \eta}\right|_{\eta=\eta^i_\lambda}$.

The particle flow particle filter (PF-PF) [9] generates proposal particles by applying the flow to particles $\{\eta^i_0\}_{i=1}^{N_p}$ distributed according to the prior. The applied flow is slightly modified from the equations described above in order to ensure that it is an invertible mapping (see Section IV and [9] for details). Application of the flow leads to the construction of a set of weighted particles $\{\eta^i_1, w^i_{n-1}\}_{i=1}^{N_p}$. The importance weight of η^i_1 is then updated to ensure that the filter is statistically consistent. $A^i(\lambda)$ and $b^i(\lambda)$ in the PF-PF (LEDH) need to be independent of the sampling process in the prior propagation phase of the i-th particle for the flow to possess the invertible mapping property. This algorithmic design requirement is reflected in the proposed algorithm described in Section IV.

IV. Particle flow particle filtering based on clustering

Since the PF-PF algorithm based on LEDH involves computation of the flow for each particle at each intermediate pseudo-time step, the computational cost can be high if we employ a reasonably large number of particles. We propose to first cluster the particles and perform flow calculation only at the cluster medoids.

We use $\gamma^{(k)}$ to denote the set containing the particle indexes that belongs to the k-th cluster and $\bar{\eta}^{(k)}$ as the medoid of the k-th cluster. Here we choose the partitioning around medoids (PAM) clustering method proposed in [10]. PAM is one of the most widely used realizations of K-medoids clustering. It iteratively chooses members from the clusters as the medoids and partitions the whole dataset into clusters around those medoids. The goal is to minimize the sum of distances between data points and the medoid in the cluster to which they belong. Compared to K-means clustering, K-medoids clustering can be used in conjunction with arbitrary distance metrics, and it is more robust to outlier data, which can arise when a few particles are located in the tails of the evolving distribution.

For each intermediate pseudo-time step, we only need to compute the flow at the K cluster medoids $\{\bar{\eta}^{(k)}\}_{k=1}^{K}$. This flow is applied to all particles belonging to the cluster. Since we are applying approximate flows, we expect that the clustering procedure will lead to a reduction in accuracy and we are interested in the tradeoff between computation and accuracy. Pseudocode for the proposed algorithms is presented in Algorithm 1. The computational complexity of the clustered PF-PF based on the LEDH is $O(N_\lambda K + N_p)$, without considering the computational complexity of the chosen clustering algorithm. Different clustering algorithms or distance metrics can be used in Line 11 of Algorithm 1, as discussed in the subsections below.

A. Euclidean distances between the states

An intuitive approach is to perform clustering based on Euclidean distances between $\{\bar{\eta}^i\}_{i=1}^{N_p}$, where $\bar{\eta}^i = g(x^i_{n-1}, 0)$ is the i-th particle propagated from x^i_{n-1} using the dynamic model without noise.

The clustering is performed using $\{\bar{\eta}^i\}_{i=1}^{N_p}$ instead of $\{\eta^i_0\}_{i=1}^{N_p}$ (particles generated with the dynamic noise). This is because we need $A^i(\lambda)$ and $b^i(\lambda)$ to be independent of the process noise. This allows us to ensure that the i-th proposed particle is derived by sampling from the prior (η^i_0) and then

Algorithm 1: Clustered PF-PF based on LEDH.

1: Initialization: Draw $\{x_0^i\}_{i=1}^{N_p}$ from the prior $p_0(x)$;
2: Set $\{w_0^i\}_{i=1}^{N_p} = \frac{1}{N_p}$;
3: **for** n = 1 to T **do**
4: Estimate P using the sample mean and the sample covariance matrix, EKF, or UKF;
5: **for** $i = 1, \ldots, N_p$ **do**
6: Calculate $\bar{\eta}^i = g(x_{n-1}^i, 0)$;
7: Propagate particles $\eta_0^i = g(x_{n-1}^i, v_n)$;
8: Set $\eta_1^i = \eta_0^i$;
9: **end for**
10: Set $\lambda = 0$;
11: Clustering: generate $\{\gamma^{(k)}, \bar{\eta}^{(k)}\}_{k=1}^{K}$ using Algorithm 2;
12: **for** $m = 1, \ldots, N_\lambda$ **do**
13: Set $\lambda = \lambda + \Delta\lambda(m)$;
14: **for** $k = 1, \ldots, K$ **do**
15: Calculate $A^{(k)}(\lambda)$ and $b^{(k)}(\lambda)$ from (6) and (7) with the linearization being performed at $\bar{\eta}^{(k)}$;
16: Migrate $\bar{\eta}^{(k)}$: $\bar{\eta}^{(k)} = \bar{\eta}^{(k)} + \Delta\lambda(j)(A^{(k)}(\lambda)\bar{\eta}^{(k)} + b^{(k)}(\lambda))$;
17: **for** $i \in \gamma^{(k)}$ **do**
18: Migrate particles: $\eta_1^i = \eta_1^i + \Delta\lambda(j)(A^{(k)}(\lambda)\eta_1^i + b^{(k)}(\lambda))$;
19: **end for**
20: **end for**
21: **end for**
22: **for** $i = 1, \ldots, N_p$ **do**
23: Set $x_n^i = \eta_1^i$;
24: $w_n^i = \dfrac{p(x_n^i|x_{n-1}^i)p(z_n|x_n^i)}{p(\eta_0^i|x_{n-1}^i)} w_{n-1}^i$;
25: **end for**
26: **for** $i = 1, \ldots, N_p$ **do**
27: Normalize $w_n^i = w_n^i / \sum_{s=1}^{N_p} w_n^s$;
28: **end for**
29: Estimate $\hat{x}_n$ from $\{x_n^i, w_n^i\}$;
30: (Optional) Resample $\{x_n^i, w_n^i\}_{i=1}^{N_p}$ and regularize to obtain $\{x_n^i, \frac{1}{N_p}\}_{i=1}^{N_p}$;
31: **end for**

applying a deterministic, invertible mapping to generate η_1^i. This also requires a relatively mild constraint on $H^i(\lambda)$ (see [9] for details). The invertible mapping enables efficient computation of the importance weights.

If some state components do not contribute to the flow parameters, we can ignore those state components when clustering. For example, in the simulation presented in the Section V, the observations are a function of the position elements; as a result the flow applied to a particle is independent of the velocity components of the states. If the state vector consists of states with different units, a normalization procedure is first performed so that the ranges of values in each dimension are the same.

B. Pearson correlation coefficient between initial flows

It is possible that although two particles are relatively close to each other initially, their flows are quite different due to a peaky likelihood function. Thus, we also consider performing clustering based on the Pearson correlation coefficients between the initially calculated flows $\zeta^i = \zeta(\eta_0^i, 0) = A^i(0)\eta_0^i + b^i(0)$. Flow vectors for all particles are computed only once, at the beginning of the particle flow, with the sole purpose of clustering the particles. We denote the Pearson correlation coefficient between ζ^i and ζ^j by p_{ij}, and use $(1 - p_{ij}) \in [0, 2]$ as the distance, because a larger value of $p_{i,j}$ indicates that the initial values ζ^i and ζ^j are more positively correlated, and this should correspond to a smaller distance.

The algorithm for this distance is a special case of Algorithm 2, with α set to 0 in Equation (8).

C. Mixed distance metric

Denote the Euclidean distance between $\bar{\eta}^i$ and $\bar{\eta}^j$ by d_{ij}. We normalize the d_{ij} values so that the maximum Euclidean distance between any two particles is 2, and the minimum value is 0. The resultant $\tilde{d}_{ij}$ values have the same range as the $1 - p_{ij}$ values.

The weighted distance metric ϕ_{ij} is given by

$$\phi_{ij} = \alpha\tilde{d}_{ij} + (1 - \alpha)(1 - p_{ij}), \tag{8}$$

where $\alpha \in [0, 1]$ is a scalar weight. The clustering algorithm with this mixed distance is presented in Algorithm 2. Its computational complexity is $O(KN_p^2)$. Although the complexity is quadratic with respect to N_p, the distance metric is only needed to be calculated once between Line 15 and Line 21 of Algorithm 2, and the K-medoids algorithm can be performed very efficiently. So the computational cost added from the clustering algorithm is relatively small. This clustering algorithm is used in line 11 of Algorithm 1.

Algorithm 2: Clustering based on the mixed distance metric. (Replaces line 11 of Algorithm 1).

11: **for** $i = 1, \ldots, N_p$ **do**
12: Calculate $A^i(\lambda)$ and $b^i(\lambda)$ from (6) and (7) with the linearization being performed at $\bar{\eta}^i$;
13: Calculate the slope $\zeta^i = (A^i(\lambda)\bar{\eta}^i + b^i(\lambda))$;
14: **end for**
15: **for** $i = 1, \ldots, N_p$ **do**
16: **for** $j = 1, \ldots, N_p$ **do**
17: **if** $j \neq i$ **then**
18: Calculate d_{ij} from $\bar{\eta}_i$ and $\bar{\eta}_j$; p_{ij} from ζ^i and ζ^j;
19: **end if**
20: **end for**
21: **end for**
22: Normalize $\{d_{ij}\}_{i\in 1:N_p, j\in 1:N_p}$ to obtain $\{\tilde{d}_{ij}\}_{i\in 1:N_p, j\in 1:N_p}$; calculate $\{\phi_{ij}\}_{i\in 1:N_p, j\in 1:N_p}$ using Equation (8);
23: K-medoids clustering based on $\{\phi_{ij}\}_{i\in 1:N_p, j\in 1:N_p}$: generate $\{\gamma^{(k)}, \bar{\eta}^{(k)}\}_{k=1}^{K}$;

V. Simulations and Results

A. Simulation setup

We adapt a multi-target acoustic tracking simulation setup proposed in [11] to create a scenario with a relatively large state space and highly informative measurements. The size of the monitored area is 40 m $\times$40 m, with $S = 25$ acoustic amplitude sensors being deployed evenly at the intersections of the grid dividing the region, as shown in Figure 1.

$C = 4$ targets move independently according to a constant velocity model $x_k^{(c)} = Fx_{k-1}^{(c)} + v_k^{(c)}$, where $x_k^{(c)} = [\mathrm{x}_k^c, \mathrm{y}_k^c, \dot{x}_k^c, \dot{y}_k^c]$ containing the x-y position and x-y velocity components of the c-th target. $F = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ is the transition matrix. $v_k^{(c)} \sim N(0, \sigma_v^2 V)$ is the process noise. To simulate different trajectories, we add noise terms sampled from $\frac{1}{20}\begin{bmatrix} 1/3 & 0 & 0.5 & 0 \\ 0 & 1/3 & 0 & 0.5 \\ 0.5 & 0 & 1 & 0 \\ 0 & 0.5 & 0 & 1 \end{bmatrix}$.

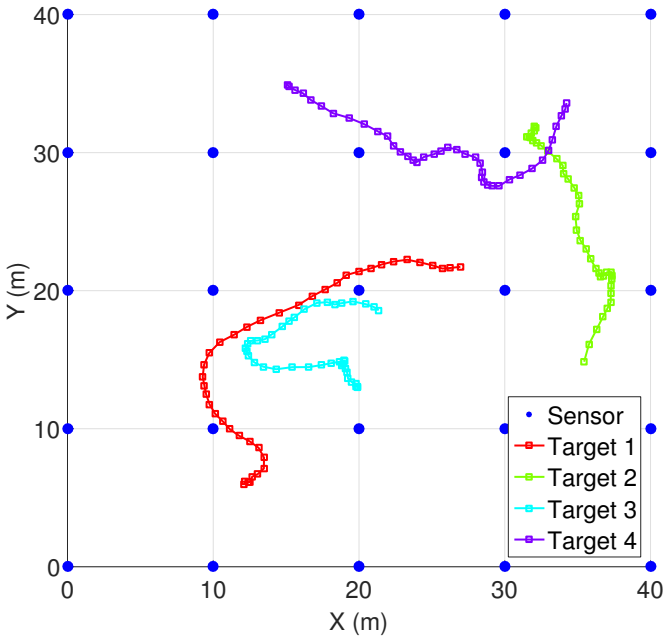

Fig. 1. The experiment setup illustrating sensor positions and target trajectories in one simulation trial.

Sensor s receives attenuated amplitudes sent out from each target. The noise-free measurement function is additive:

$$\bar{z}^s(x_k) = \sum_{c=1}^{C} \frac{\Psi}{||(\mathrm{x}_k^c, \mathrm{y}_k^c)^T - R^s||^\epsilon + d_0}. \tag{9}$$

We set $\Psi = 10, \epsilon = 1, d_0 = 0.1$. For sensor s, the noisy measurement z_k^s is tainted with an additive Gaussian noise:

$$z_k^s = \bar{z}^s(x_k) + w_k^s$$

where $x_k = (x_k^{(1)}; \ldots; x_k^{(C)})$ is a 16-component state vector, $w_k^s \sim N(0, \sigma_w^2)$. σ_w^2 is set to 0.01, thus highly informative measurements are generated.

The initial states of the four targets are $[12, 6, 0.001, 0.001]^T$, $[32, 32, -0.001, -0.005]^T$, $[20, 13, -0.1, 0.01]^T$ and $[15, 35, 0.002, 0.002]^T$, respectively. 100 set of trajectories are simulated. Each of them generate one measurement. All filtering algorithms are executed 5 times on the same measurement, with different initialized states. The reported results thus reflect the outcome of 500 trials, varying either the target trajectories or the filter initializations. Matlab is used to implement the simulation.

B. Compared filtering algorithms and parameter values

We compare the performance of the proposed algorithm with the EDH filter [3], the LEDH filter [4], the Gaussian particle flow importance sampling particle filter [6], bootstrap particle filters (BPF), and the extended Kalman filter (EKF).

We perform particle flow calculations at exponentially spaced discrete steps as recommended in [12]. The constant ratio between step sizes is set to 1.2, i.e. $q = \frac{\Delta\lambda(j)}{\Delta\lambda(j-1)} = 1.2$, for $j = 2, 3, \ldots, N_\lambda$. $N_\lambda = 29$ steps are used. $\bar{\eta}_0$ and P are estimated using an EKF executed in parallel to the particle flow.

In each simulation, particles are sampled for all algorithms from the same initial distribution. The mean of the initial distribution is sampled from a large variance Gaussian centered around the true initial state. If the initial mean is located outside the surveillance sensor grid, it is rejected and resampled. The variance for the position elements is set to 100, and the variance for the velocity elements is set to 1. We set $\sigma_v^2 = 0.05$ and $V = \begin{bmatrix} 3 & 0 & 0.1 & 0 \\ 0 & 3 & 0 & 0.1 \\ 0.1 & 0 & 0.03 & 0 \\ 0 & 0.1 & 0 & 0.03 \end{bmatrix}$, to specify the process noise. Resampling and regularization are performed when the effective sample size (ESS) is less than $\frac{N_p}{2}$. All algorithms use 500 particles except the two bootstrap particle filters which use, respectively, 10^5 and 10^6 particles.

C. Tracking performance

Since the number of targets in our simulation is fixed, we use the optimal mass transfer (OMAT) metric [13] to compare and evaluate the algorithms. The p-th order OMAT metric $d_p(X, \hat{X})$ is defined as

$$d_p(X, \hat{X}) = (\frac{1}{C} \min_{\pi \in \Pi} \sum_{c=1}^{C} d(x_c, \hat{x}_{\pi(c)})^p)^{1/p} \tag{10}$$

where $X = \{x_1, x_2, \ldots, x_C\}$ and $\hat{X} = \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_C\}$ are arbitrary sets. Π is the set of possible permutations of $\{1, 2, \ldots, C\}$. The scalar p is a fixed parameter and $d(x, \hat{x})$ is the Euclidean distance between x and $\hat{X}$. We set p to 1, so the OMAT metric minimizes the Euclidean distance between the true target positions and the permutation of estimated target positions.

We first demonstrate through Figure 2 that the PF-PF (LEDH) has the best tracking accuracy compared with other tested filtering algorithms. The PF-PF (LEDH) with 500 particles outperforms the BPF with 1 million particles in terms of tracking accuracy.

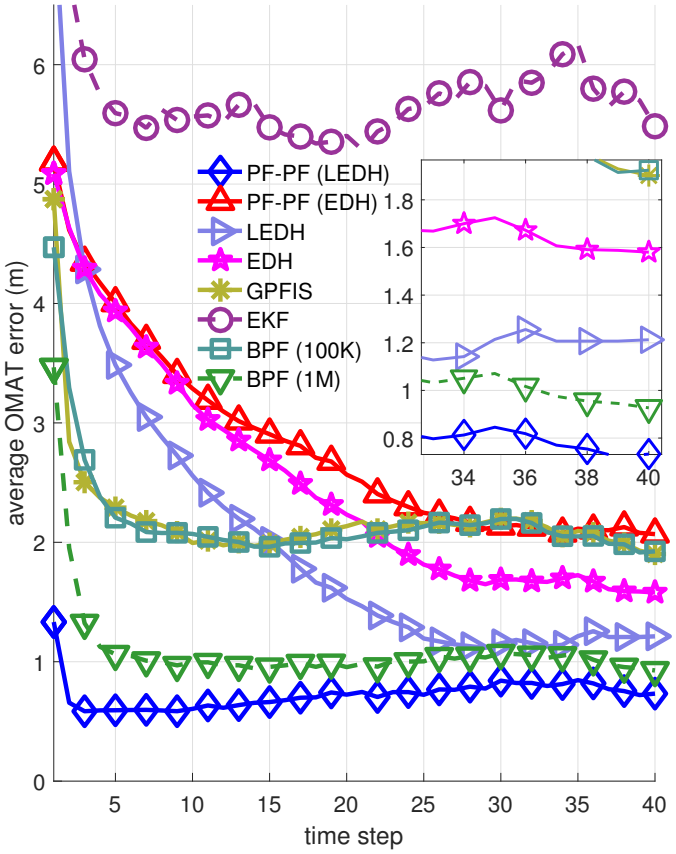

Fig. 2. Average errors at each time step from different filtering algorithms.

We now examine the impact of clustering. We conducted tests using different values of α in the mixed clustering distance and observed almost the same performance for $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$. Thus, we fix the value of the weight α to 0.25 for the rest of the paper. This means that we place more importance on the correlation between the initial flow vectors in evaluating the clustering distance. One example of the estimated tracks is shown in Figure 3. The estimated trajectories closely follow the true trajectories for most of the tracking period.

To investigate the impact of the number of clusters used in K-medoids clustering, we choose $K \in \{30, 100, 200, 300\}$. The average OMAT errors at each time step are shown in Figure 4. We observe that PF-PF (LEDH) with $K \in \{100, 200, 300\}$ clusters have slightly higher average errors than PF-PF (LEDH), but the values are comparable. The performance deteriorates slightly as the number of clusters is reduced. Even with 100 clusters, the tracking performance is better than the LEDH or GPFIS filters. The error becomes significantly larger when the number of clusters is reduced to 30.

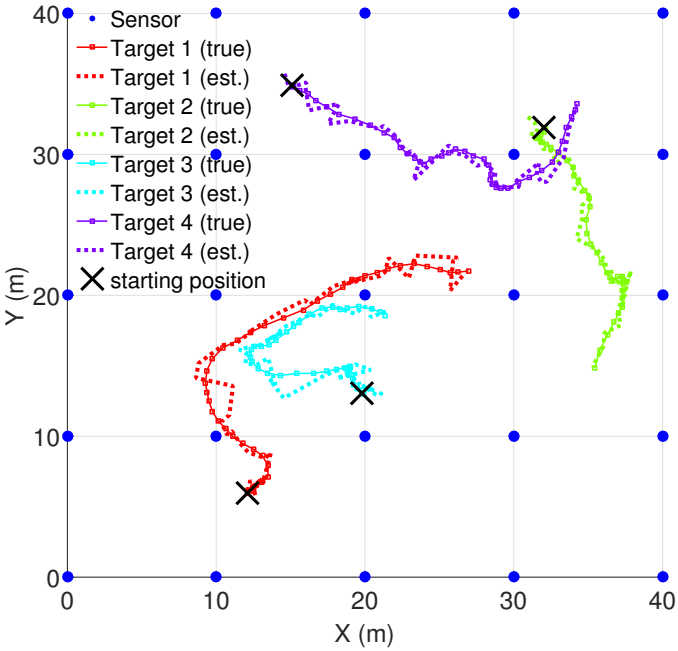

Fig. 3. One example of estimated trajectories using PF-PF (LEDH) with 100 clusters and $\alpha = 0.25$. The crosses indicate the starting positions of the four targets and the solid lines show their true trajectories. Dotted lines show the estimated trajectories, which are close to the true trajectories hence there is considerable overlap between the dotted and solid lines.

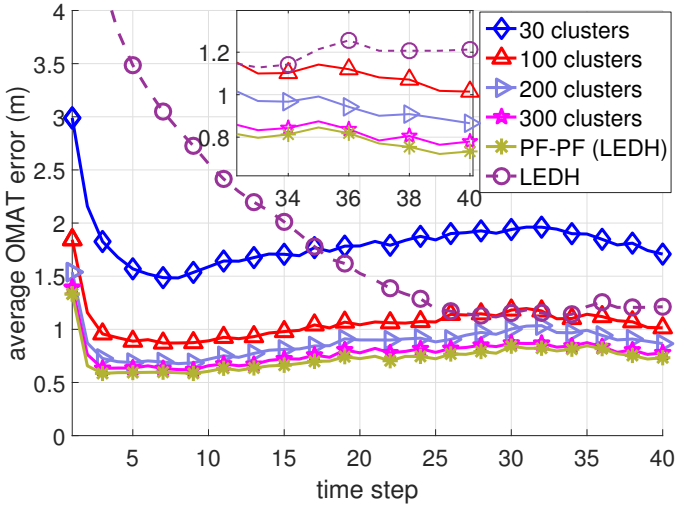

Fig. 4. Average errors at each time step from PF-PF (LEDH) and its clustering variants with different number of clusters.

The main motivation for using clustering is to reduce the computational cost of the PF-PF (LEDH) algorithm. To examine the trade-off between the tracking performance and the execution time, boxplots of average OMAT errors from PF-PF (LEDH) and its clustering-based variants are plotted against their execution time per step (Figure 5). From the bottom figure we can see that with 100 clusters or more, PF-PF (LEDH) via clustering has a similar median, as well as the first and third quartiles, compared with PF-PF (LEDH). However, with 100 clusters, the average execution time per

time step is one quarter of that of the PF-PF (LEDH) without clustering, using the same number $(N_p = 500)$ of particles. The number of outliers, shown in the top figure of Figure 5, increases when the number of clusters decreases, especially in the region where the average OMAT error is larger than 6 meters. This shows that the computational saving from a reduced number of clusters comes at a cost of increasing the chance of a poor tracking outcome.

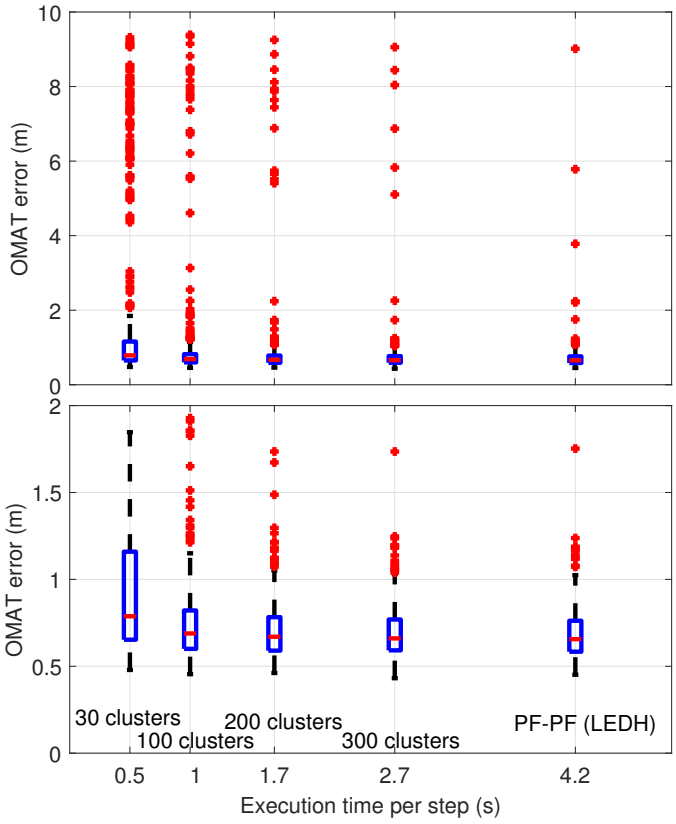

Fig. 5. Boxplots of average OMAT errors, plotted against the average execution time per time step, produced with an Intel Xeon E5-4650 2.70GHz CPU. The top figure shows the full boxplots; the bottom one concentrates on the region where the OMAT error is between 0 and 2. All algorithms use 500 particles.

VI. Conclusion

In this paper, we explored using clustering algorithms to expedite the particle flow particle filtering algorithm. We designed a mixed distance metric to perform clustering of particles, while retaining the invertible mapping property of the deterministic flow, which allows efficient calculation of importance weights.

We show that by clustering particles and perform the expensive flow calculations only at the cluster medoids, we can significantly reduce the computational cost with a small sacrifice in accuracy. In future research we will explore the design of an adaptive method to choose and vary the number of clusters over time.

References

[1] T. Bengtsson, P. Bickel, and B. Li, "Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems," in *Probability and Statistics: Essays in Honor of David A. Freedman*, D. Nolan and T. Speed, Eds. Beachwood, OH, USA: Institute of Mathematical Statistics, 2008, vol. 2, pp. 316–334.

[2] F. Daum and J. Huang, "Nonlinear filters with log-homotopy," in *Proc. SPIE Signal and Data Processing of Small Targets*, San Diego, CA, USA, Sep. 2007, p. 669918.

[3] F. Daum, J. Huang, and A. Noushin, "Exact particle flow for nonlinear filters," in *Proc. SPIE Conf. Signal Proc., Sensor Fusion, Target Recog.*, Orlando, FL, USA, Apr. 2010, p. 769704.

[4] T. Ding and M. J. Coates, "Implementation of the Daum-Huang exact-flow particle filter," in *Proc. IEEE Statistical Signal Processing Workshop (SSP)*, Ann Arbor, MI, USA, Aug. 2012, pp. 257–260.

[5] V. Maroulas, K. Kang, I. D. Schizas, and M. W. Berry, "A learning drift homotopy particle filter," in *Proc. Intl. Conf. on Information Fusion*, Washington, DC, July 2015, pp. 1930–1937.

[6] P. Bunch and S. Godsill, "Approximations of the optimal importance density using Gaussian particle flow importance sampling," *J. Amer. Statist. Assoc.*, 2015, accepted, see http://arxiv.org/abs/1406.3183.

[7] S. Reich, "A guided sequential Monte Carlo method for the assimilation of data into stochastic dynamical systems," in *Recent Trends in Dynamical Systems*. Springer Basel, 2013, vol. 35, pp. 205–220.

[8] F. E. de Melo, S. Maskell, M. Fasiolo, and F. Daum, "Stochastic particle flow for nonlinear high-dimensional filtering problems," *arXiv preprint arXiv:1511.01448*, 2015.

[9] Y. Li, L. Zhao, and M. J. Coates, "Particle flow for particle filtering," in *Proc. Intl. Conf. Acoustics, Speech and Signal Proc. (ICASSP)*, Shanghai, China, Mar. 2016, accepted.

[10] L. Kaufman and P. J. Rousseeuw, "Finding groups in data: An introduction to cluster analysis," *Wiley Series in Probability and Mathematical Statistics. Applied Probability and Statistics*, 1990.

[11] O. Hlinka, O. Sluciak, F. Hlawatsch, P. M. Djuric, and M. Rupp, "Distributed Gaussian particle filtering using likelihood consensus," in *Proc. Intl. Conf. Acoustics, Speech and Signal Proc. (ICASSP)*, Prague, Czech Republic, May 2011, pp. 3756–3759.

[12] F. Daum and J. Huang, "Particle flow with non-zero diffusion for nonlinear filters," in *Proc. SPIE Conf. Signal Proc., Sensor Fusion, Target Recog.*, Baltimore, MD, USA, May 2013, p. 87450P.

[13] D. Schuhmacher, B. T. Vo, and B. N. Vo, "A consistent metric for performance evaluation of multi-object filters," *IEEE Trans. Signal Process.*, vol. 56, no. 8, pp. 3447–3457, Aug 2008.