

# Distributed particle filter for bearing-only tracking

Jun Ye Yu

April 4, 2018

## 1 Introduction

In this report we present four distributed particle filters for single-target bearing-only tracking. The first filter factorizes the joint log-likelihood function using six sufficient statistics that can be computed using distributed summation. The second filter uses likelihood consensus to encode the particle log-likelihoods using a small number of basis functions. The third filter constructs a graph over all particles and the Eigenvectors of the resulting Laplacian matrix are used to encode the particle log-likelihood using a minimal number of coefficients. Finally, the fourth filter groups all particles into clusters and computes the cluster joint likelihood. The individual particle weights are then recovered via convex minimization. For the remainder of the report, we refer to the four particle filters as **CSSpf** [1], **LCpf** [2], **LApf** [3] and **Clusterpf** [4] respectively. We also include the centralized *bootstrap particle filter* (**BSpf**) as baseline.

The remainder of the report is organized as follows. Sec. 2 defines the tracking problem. Sec. 3 presents the particle filters. Sec. 4 compares the filters' performance and Sec. 6 concludes the report.

## 2 Problem statement

A network of  $S$  sensors collaboratively track a single moving target over time. The sensors have fixed position  $[x_s, y_s]$ ,  $s = 1 \dots S$ . The target state at time  $k$  is modeled as  $X(k) = [x_t(k), y_t(k), \dot{x}_t(k), \dot{y}_t(k)]$  where  $x_t(k)$ ,  $y_t(k)$  are the target position and  $\dot{x}_t(k)$ ,  $\dot{y}_t(k)$  are its velocity.

At time  $k$ , the target transitions to new state  $X(k)$  with probability  $f(X(k)|X(k-1))$  which depends on the target dynamic model. Each sensor  $s$  also receives a noisy measurement  $z_s(k)$  with likelihood  $f(z_s(k)|H_s(X(k)))$  where  $H_s(\cdot)$  is the (possibly sensor-dependent) measurement function. The sensors have unity target detection probability and receive no clutter measurement.

The objective is to estimate the posterior target density  $p(X(k)|z_1(k), \dots, z_S(k))$  at each time step  $k$ .

## 3 Distributed particle filters for bearing-only tracking

In a particle filter, the posterior target density is modeled using a set of  $N$  particles with normalized weights  $\{X_i(k), w_i(k)\}_{i=1}^N$ , and the objective is to recursively estimate the posterior particle weights. This in turn requires the computation of joint log-likelihood:

$$w_i(k) \propto \log(f(z_1(k), \dots, z_S(k)|X_i(k))) = \sum_{s=1}^S \log(f(z_s(k)|X_i(k))) \quad (1)$$

where measurements from different sensors are considered to be conditionally independent.

For the remainder of this section, we present four distributed particle filters which compute the joint log-likelihood in different manners. We omit time step indice  $k$  where there is no ambiguity. For convenience of notation, let  $\gamma_s = [\log(f(z_s|X_1)), \dots, \log(f(z_s|X_N))]^T$  denote the column vector of the  $N$  particle log-likelihoods at sensor  $s$ . Similarly, let  $\gamma = [\log(f(z_1, \dots, z_S|X_1)), \dots, \log(f(z_1, \dots, z_S|X_N))]^T$  denote the column vector of joint log-likelihood.

### 3.1 Constraint sufficient statistics particle filter

In the CSSpf, the likelihood function is approximated as follows [1]

$$\log(f(z_s|X)) \approx \sum_{j=1}^6 G_{s,j} F_j(X) \quad (2)$$

$$\begin{aligned} G_{s,1} &= (Z_{s,\theta})^2 / R_\theta & F_1(X) &= 1 \\ G_{s,2} &= \cos^2(Z_{s,\theta}) / R_\theta & F_2(X) &= x_t^2 \\ G_{s,3} &= \sin^2(Z_{s,\theta}) / R_\theta & F_3(X) &= y_t^2 \\ G_{s,4} &= \sin(Z_{s,\theta}) \cos(Z_{s,\theta}) / R_\theta & F_4(X) &= -2x_t y_t \\ G_{s,5} &= Z_{s,\theta} \cos(Z_{s,\theta}) / R_\theta & F_5(X) &= 2x_t \\ G_{s,6} &= Z_{s,\theta} \sin(Z_{s,\theta}) / R_\theta & F_6(X) &= -2y_t \\ Z_{s,\theta} &= y_s \sin(z_s) - x_s \cos(z_s) \\ R_\theta &= E((x_t - x_s)^2 + (y_t - y_s)^2) (1 - \exp^{-2\sigma_\theta^2}) / 2 \end{aligned}$$

where the expectation term in  $R_\theta(\cdot)$  is taken over all particles  $X_i$ . The functions  $F_j(X)$  depend only on target state  $X$  and are known to all sensors. The sufficient statistics  $G_{s,j}$  depend only on local information from sensor  $s$ . In other words, we approximate the log-likelihood function by the weighted combination of six basis functions  $F_j(X)$  with corresponding weight coefficients  $G_{s,j}$ .

This formulation leads to the following approximate joint log-likelihood function

$$\log(f(z_1, \dots, z_S|X)) \approx \sum_{j=1}^6 F_j(X) \left( \sum_{s=1}^S G_{s,j} \right) \quad (3)$$

where the summation terms  $\left( \sum_{s=1}^S G_{s,j} \right)$  can be interpreted as the global sufficient statistics. These global sufficient statistics can be computed in a distributed manner by running six consensus algorithms in parallel. The six basis functions of CSSpf are specifically tailored for bearing-only tracking. For other measurement model, re-derivation of the filter is required.

### 3.2 Likelihood consensus particle filter

In LCpf, we approximate the log-likelihood function as follows:

$$\log(f(z_s|X)) \approx \sum_{j=1}^J \alpha_{s,j} \beta_j(X) \quad (4)$$

where  $\beta_j(X)$  is the  $j^{th}$  sensor-independent basis function and  $\alpha_{s,j}$  is the corresponding coefficient that encompasses all the local information of sensor  $s$ . We note that the CSSpf can be seen as a special case of LCpf and that  $\alpha_{s,j}$  is analogous to  $G_{s,j}$  and  $\beta_j(X)$  is analogous to  $F_j(X)$ .

For each sensor  $s$ , we construct the column vector  $\gamma_s = [\log(f(z_s|X_1)), \dots, \log(f(z_s|X_N))]^T$  where  $T$  denotes the transpose operation. Given the  $N$  particles  $X_i$ , we construct the  $N \times J$  matrix  $\Phi$  as follows:

$$\Phi = \begin{pmatrix} \beta_1(X_1) & \dots & \beta_J(X_1) \\ \vdots & \ddots & \vdots \\ \beta_1(X_N) & \dots & \beta_J(X_N) \end{pmatrix} \quad (5)$$

We seek a set of coefficients  $\alpha_s = [\alpha_{s,1}, \dots, \alpha_{s,J}]^T$  such that the approximation error  $\gamma_s - \Phi\alpha_s$  is minimized. Using the least-square approach yields the coefficients vector

$$\alpha_s = (\Phi^T \Phi)^{-1} \Phi^T \gamma_s \quad (6)$$

As in the case of CSSpf, computing the joint log-likelihood amounts to running  $J$  consensus algorithms in parallel and computing  $\sum_{s=1}^S \alpha_{s,j}, j = 1 \dots J$ .

### 3.3 Laplacian approximation particle filter

In LApf, we consider each particle  $X_i$  a vertex on a graph. The *Delaunay triangulation* (DT) is used to generate the graph edges. The resulting Laplacian matrix is used to construct a transformation that encodes particle log-likelihoods using a minimal number of coefficients.

Let  $L$  denote the Laplacian matrix of the DT graph. The eigenvectors of  $L$  are used to transform particle log-likelihoods into Laplacian domain. Using all  $N$  eigenvectors is obviously counterproductive since we incur the computational overhead of eigendecomposition and achieve no reduction in communication overhead (i.e., we still have to aggregate  $N$  coefficients).

Assume that  $m \leq N$  eigenvectors are used as the basis of transformation and let  $E_m$  denote the resulting matrix where each column is an eigenvector. We compute the local coefficients at sensor  $s$  as follows:

$$\alpha_s = E_m^T \gamma_s \quad (7)$$

The global coefficients are the summation of local coefficients across all  $S$  sensors:  $\alpha = \sum_s \alpha_s$ . Finally, the approximate joint log-likelihood can be computed as follows:

$$\hat{\gamma} = E_m \alpha = E_m \sum_s \alpha_s \quad (8)$$

Since the particle log-likelihoods can be considered as a smooth signal over the graph (i.e., particles close to each other have similar log-likelihoods), most of their energy should be concentrated in the coefficients corresponding to lower frequency basis vectors. In other words, we should retain the  $m$  eigenvectors corresponding to the  $m$  smallest eigenvalues.

We note that LApf is similar to CSSpf in that both filters encode the particle log-likelihoods directly using a minimal number of coefficients. However, for LApf, all sensors must be synchronized so that they have the same particles; otherwise they would obtain a different particle graph and by extension different eigenvectors for the encoding. The CSSpf has no such restrictions.

### 3.4 Clustering particle filter

In Clusterpf, the particles are grouped into  $C$  clusters based on their position. The sensors reach consensus on the cluster log-likelihoods rather than individual particle log-likelihoods. For  $C \ll N$ , significant reduction in communication overhead can be achieved.

We follow the approach in [2]. The log-likelihood of each cluster is equal to the aggregate log-likelihoods of its constituent particles. Let  $\gamma^c$  denote the joint log-likelihood of the clusters after consensus. Let  $A_C$  denote the  $C \times N$  cluster assignment matrix where  $A_C(i, j) = 1$  if particle  $j$  belongs to cluster  $i$ .

In order to recover the individual particle joint log-likelihoods  $\gamma$ , we again construct DT graph, compute the Laplacian matrix  $L$ , and then solve the following convex minimization problem:

$$\underset{\gamma}{\text{minimize}} \quad \gamma^T L \gamma \quad (9)$$

$$\text{subject to} \quad A_C \gamma = \gamma^c \quad (10)$$

In other words, we seek to assign particle log-likelihood values that are smooth with respect to particle proximity while ensuring the aggregate particle values are equal to the cluster value. As in the case of LApf, the Clusterpf requires that all sensors have the same particles.

### 3.5 Computational overhead

In this section we compare the computational overhead of the four filters. More specifically, we compare the overhead for particle log-likelihood computation.

Consider first CSSpf. Each sensor computes the six local sufficient statistics with complexity  $O(6N)$ . These statistics are then aggregated via distributed consensus with complexity  $O(6S * NGossip)$ . Finally, the log-likelihoods are computed at all sensors using the global sufficient statistics with complexity  $O(6S * N)$ . Thus, the overall complexity of CSSpf is thus  $O(6S * N + S * NGossip + 6S * N)$ . Since  $N > NGossip$  in general, the complexity is dominated by  $O(6S * N)$ .

Consider next LCpf. Let  $J$  denote the number of basis functions. Each sensor needs to generate a  $N \times J$  matrix to compute the local coefficients. Then  $J$  coefficients are aggregated via distributed consensus over  $NGossip$  iterations. The log-likelihoods are finally computed from the global coefficients. The overall complexity is thus  $O(S * N * J + S * NGossip * J + S * N) \subset O(S * N * J)$ .

Consider next LAPf. The Delaunay triangulation for graph construction has complexity  $O(N \log(N))$ . The eigenvalue decomposition has complexity  $O(N^3)$ . Assume  $m$  eigenvectors are used to decode the local log-likelihoods. Then  $m$  scalars are aggregated via distributed consensus. The joint log-likelihoods are then recovered from the  $m$  aggregate scalars. The overall complexity is thus  $O(S * N \log(N) + S * N^3 + S * m * N + S * m * NGossip + S * m * N) \subset O(S * N^3)$ .

Finally consider Clusterpf. Particle clustering has complexity  $O(N * C * 4 * I)$  where  $I$  is the number of clustering iterations (with default value of 100) and the constant 4 is the target state dimension.  $C$  cluster log-likelihoods are then aggregated across all sensors. We again have Delaunay triangulation for graph construction. The log-likelihood is recovered via convex minimization with complexity  $O(\sqrt{N})$ . The overall complexity is thus  $O(S * N * C * 4 * I + S * C * NGossip + S * N \log(N) + S \sqrt{N}) \subset O(S * N * C * I + S * N \log(N))$ . Since we can assume that  $C * I > \log(N)$  (i.e., for  $N = 10^4$ ,  $C = 4$ ,  $C * I > \log(N)$  for  $I > 2$ ), the complexity is dominated by  $O(S * N * C * I)$ .

Overall, CSSpf has the lowest computational overhead whereas LAPf has the highest overhead due to the eigenvalue decomposition. For  $6 < J \leq N$ , LCpf has the second lowest overhead.

## 4 Performance evaluation

### 4.1 Simulation setup

The target state evolves over time following a discrete-time model:

$$X(k+1) = F(X(k)) + \xi(k) \quad (11)$$

where  $F(X(k))$  is the dynamic model and  $\xi(k)$  is the zero-mean Gaussian process noise. The simulated target randomly switches between two different motion models: constant velocity with probability  $P_{cv} = 0.05$  and coordinated turn with probability  $1 - P_{cv} = 0.95$ .

For constant velocity, we have

$$F(X(k)) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (12)$$

For coordinated turn, we have

$$F(X(k)) = \begin{bmatrix} 1 & 0 & \frac{\sin(\Omega)}{\Omega(k)} & -\frac{1-\cos(\Omega(k))}{\Omega(k)} \\ 0 & 1 & \frac{1-\cos(\Omega(k))}{\Omega(k)} & \frac{\sin(\Omega(k))}{\Omega(k)} \\ 0 & 0 & \cos(\Omega(k)) & -\sin(\Omega(k)) \\ 0 & 0 & \sin(\Omega(k)) & \cos(\Omega(k)) \end{bmatrix} \quad (13)$$

where  $\Omega(k)$  is the turning rate

$$\Omega(k) = \frac{a}{\sqrt{\dot{x}^2(k) + \dot{y}^2(k)}} \quad (14)$$

with  $a = 0.5$  being the maneuver acceleration parameter.

All sensors receive noisy bearing measurements (in radians) from the target.

$$H_s(X(k)) = \arctan 2 \left( \frac{x_t - x_s}{y_t - y_s} \right) + \eta(k) \quad (15)$$

The process and measurement noises  $\xi(k)$  and  $\eta(k)$  have covariance matrices  $Q$  and  $R$  respectively.

$$Q = \sigma_a^2 \begin{bmatrix} \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 1 & 0 \\ 0 & \frac{1}{2} & 0 & 1 \end{bmatrix} \quad (16)$$

$$R = \sigma_\theta^2 \quad (17)$$

where  $\sigma_a = 10^{-4}$ , and  $\sigma_\theta = 0.0873$  rad = 5 degree.

## 4.2 Algorithm setup

All particle filters use a total of  $N = 500$  particles. At time step 1, we generate the initial particles using the true target state:  $X_i(1) \sim \mathcal{N}(X(1), R_{\text{initial}})$  with  $R_{\text{initial}} = \text{diag}([0.5^2, 0.5^2, 0.05^2, 0.05^2])$ .

For LCPf, we use a set of basis functions involving all permutations of  $x_t^i y_t^j$  with  $0 \leq i, j \leq d$  where  $d$  is some user-specified max degree. (i.e., For  $d = 2$ , the basis functions would be  $\beta_1(X) = x_t^0 y_t^0 = 1, \beta_2(X) = x_t^0 y_t^1 = y_t, \dots, \beta_9(X) = x_t^2 y_t^2$ .) Note that, due to our choice of basis functions, all particles must remain synchronized across all sensors as in the case of LAPf and Clusterpf. For LAPf, we construct a DT graph and retain  $m < N$  eigenvectors as the basis of Laplacian transformation. For Clusterpf, all particles are grouped into  $C$  clusters and a DT graph is constructed to recover individual particle weights.

The random number generators are synchronized to ensure that the particles remain the same across sensors. Distributed summation is performed using gossip algorithms. At each time step, we perform *NGossip* gossip iterations. At each gossip iteration, each sensor  $i$  broadcasts its local values  $G_i$ , receives broadcasts from its neighbors, and then updates its local values as a weighted aggregate:

$$G_{i,\text{new}} = w_{ii}G_{i,\text{old}} + \sum_{j \in N_i} w_{ij}G_{j,\text{old}} \quad (18)$$

$$w_{ij} = \begin{cases} \frac{1}{1+\max(d_i, d_j)} & j \in N_i \\ 1 - \sum_{j \in N_i} w_{ij} & i = j \\ 0 & j \notin N_i \end{cases} \quad (19)$$

where  $N_i$  denotes the set of neighboring sensors of sensor  $i$ ,  $d_i = |N_i|$ , and Metropolis weight is used for the update. After a total of *NGossip* iterations, a max consensus algorithm is run to ensure all sensors obtain the same values.

In our codes, we do not implement a loop for the gossip iterations. Instead we define an update matrix  $W$  where  $W(i, j) = w_{ij}$ . Therefore, given initial values  $G_{\text{initial}} = [G_1, \dots, G_S]^T$ , the final values can be easily computed as

$$G_{\text{final}} = W^{NGossip} G_{\text{initial}} \quad (20)$$

In the remainder of the section, we run a number of Monte Carlo simulations to evaluate the performance of the four filters. The track remains the same in each trial; but the measurements differ. We evaluate the algorithms' performances using the *root mean squared error* (RMSE) of position estimate and discrepancy of normalized particle weights,  $\|w_{\text{true}} - w_{\text{approx}}\|_2$ , averaged over all time steps and all trials. The first metric is self-explanatory and the second metric is meant to evaluate the impact of different filters' encoding scheme on the overall tracking performance.

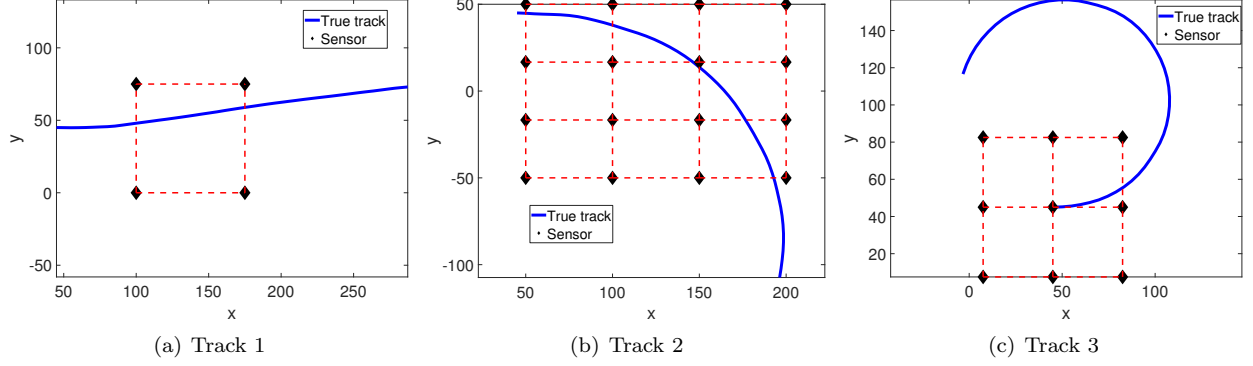


Figure 1: Target tracks (blue curve) and sensor positions (black diamond). Sensors connected by red dashed lines are within broadcast range of each other.

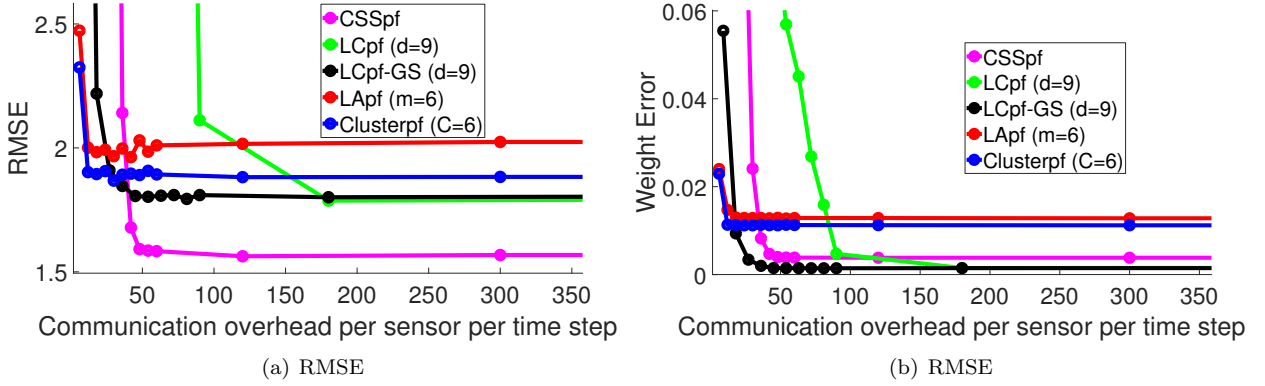


Figure 2: Average RMSE and particle weights error with respect to total communication overhead per time step for track 1 ( $N = 500$ , 200 Monte Carlo trials).

### 4.3 Performance comparison between filters

In this section, we compare the four filters against each other. We use a centralized bootstrap particle filter as baseline. Finally, we include a modified version of LCpf where we use the Gram-Schmidt process to obtain an orthonormal encoding matrix  $\Phi'$  (i.e.,  $\|\Phi'(:, i)\|_2 = 1, i = 1, \dots, J$ ). This filter is referred to as LCpf-GS. We consider three different test tracks shown in Fig. 1.

Fig. 2 shows the average RMSE and weight error with respect to communication overhead per sensor per time step for all algorithms for track 1. For overhead  $< 25$ , Clusterpf and LAPf have the lowest RMSE. Note that these two filters have low RMSE even at  $NGossip = 1$  (i.e., communication overhead =  $1 \times 6 = 6$ ). For overhead  $> 50$ , CSSpf has the best performance by a significant margin. The LCpf requires the highest communication overhead to achieve adequate tracking performance. Consider next the weight error curves. For all filters, we can notice a correspondence between weight error and RMSE (i.e., higher error equals higher RMSE) as expected; but two filters having the same weight error do not necessarily have the same RMSE. For instance, for overhead = 100, LCpf has low weight error on-par with that of CSSpf, yet its RMSE is significantly higher.

Fig. 3 shows the tracking results for track 2. For overhead  $< 200$ , LAPf and Clusterpf have the lowest RMSE. The LCpf-GS achieves similar RMSE as LAPf once the overhead exceeds 200. For LCpf, over 100 Gossip iterations are required to achieve adequate tracking performance. Again, for each filter, there is a clear correspondence between RMSE and weight error.

Fig. 4 shows the tracking results for track 3. Again, LAPf and Clusterpf achieve robust tracking perfor-

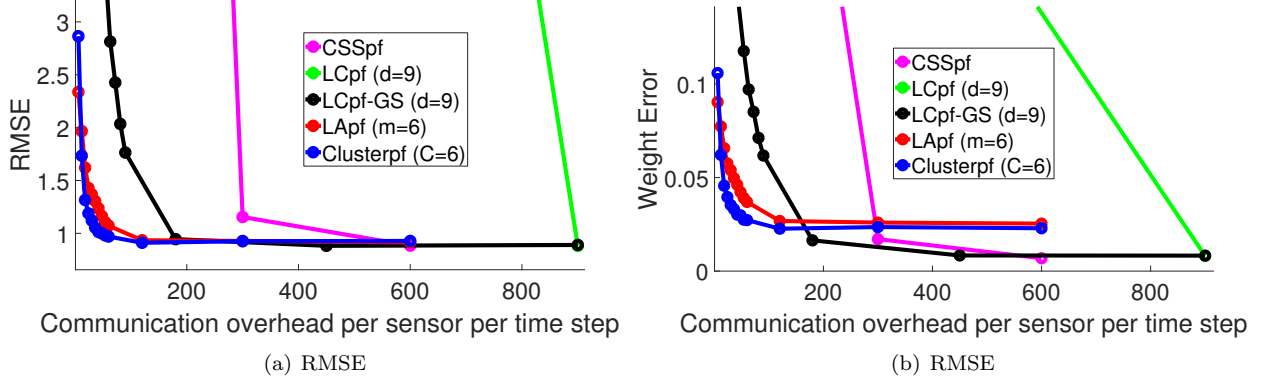


Figure 3: Average RMSE and particle weights error with respect to total communication overhead per time step for track 2 ( $N = 500$ , 200 Monte Carlo trials).

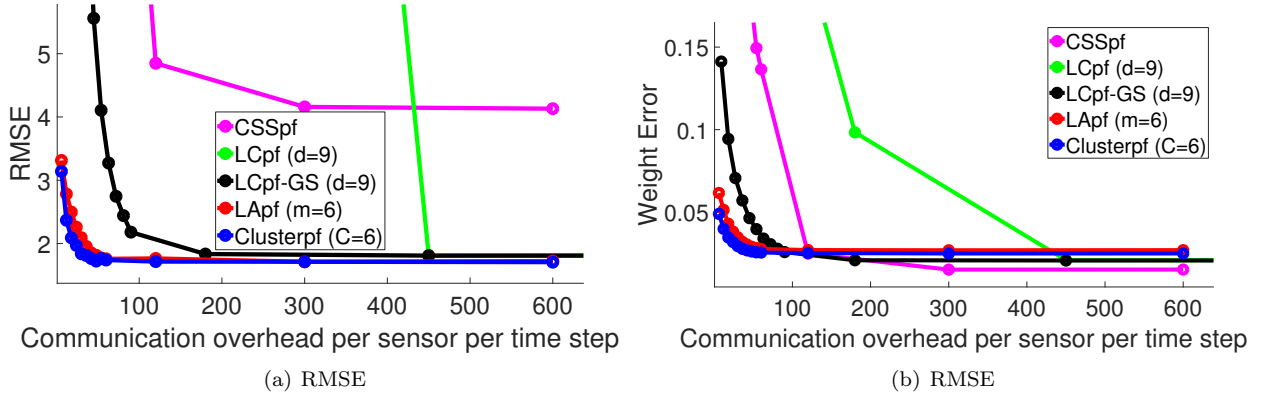


Figure 4: Average RMSE and particle weights error with respect to total communication overhead per time step for track 3 ( $N = 500$ , 200 Monte Carlo trials).

mance even with very low communication overhead. LCpf achieves good performance for  $NGossip \geq 50$ . CSSpf has poor performance even for  $NGossip \geq 100$  even though its weight error is lowest for  $NGossip \geq 30$ .

In all three test tracks, the LAPf and Clusterpf yield robust tracking performance even with very low communication overhead. The other filters may outperform LAPf and Clusterpf albeit at the cost of higher overhead. In particular, the LCpf requires significantly higher communication overhead to achieve similar RMSE as the other filters.

Fig. 5 shows the average runtime of the filters. We omit the results for tracks 1 and 2 since the trends are similar. The LAPf has the highest runtime due to the overhead of eigendecomposition. The Clusterpf has the second highest runtime. The other filters have similar low runtime. We note that increasing  $NGossip$  does not increase total runtime due to our implementation of the gossip algorithms.

#### 4.4 Sensitivity analysis of particle filters

Simulation results in the previous sections suggest that LAPf and LCpf are considerably less susceptible to gossiping noise than the other three filters. In this section, we conduct a sensitivity analysis of all filters.

We run a centralized BSpf for all three tracks. At each time step, we re-compute the posterior particle weights using all distributed filters. We then compute and report the discrepancy of approximate and exact particle weights (i.e.,  $\|W_{BS} - W_{CSS}\|_2$ ).

The goal is to understand each filter's susceptibility to gossiping error. The aggregate values are first

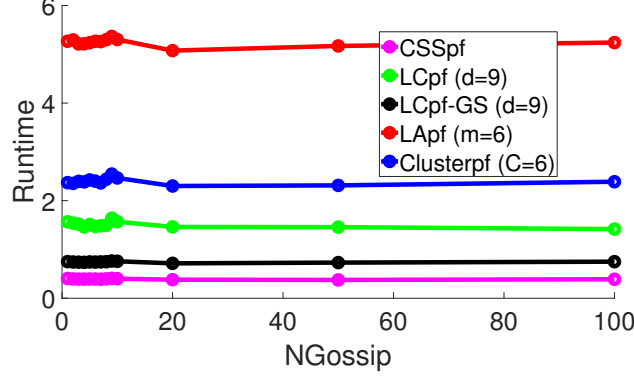


Figure 5: Average total runtime for track 3

calculated exactly without gossiping; then we inject perturbation into the aggregate values. Let  $\alpha$  denote the true aggregate value and  $x$  the perturbation level, then the final aggregate value used in weight computation is  $\alpha(1 + x/100)$  or  $\alpha(1 - x/100)$  with equal probability. Note that, in this definition, perturbation level is equal to the AER. We choose this approach so we can control the AER exactly without fine-tuning the number of gossip iterations.

We repeat this test but run a fixed number of gossip iterations for all filters. The perturbations in this case are generated from gossiping errors.

Fig. 6 shows the average weight discrepancy with respect to perturbation level and number of gossip iterations. In all three tracks, LCpf has the worst performance by far followed by CSSpf. Even at very low perturbation level, the weight discrepancy of LCpf can go above 0.2. On the other hand, LCpf-GS and LApf have consistently the best performance with LApf having an edge for perturbation  $> 100$ . And we note that, for all values of *NGossip*, the weight discrepancy of LApf and Clusterpf never exceeds 0.2.

## 5 Theoretical analysis

All four filters encode the particle log-likelihoods in a similar fashion but yet display drastically different performance at low communication overhead.

Let  $\gamma_s$  denote the  $N \times 1$  column vector of particle log-likelihoods at sensor  $s$  and let  $\gamma = \sum_{s=1}^S \gamma_s$  denote the joint log-likelihoods. Denote by  $\Psi$  the  $N \times J$  encoding matrix where  $J \leq N$  is the number of encoding coefficients. In general,  $\Psi$  is a function of local particles  $x_i$ . Finally, denote by  $\alpha_s$  the  $J \times 1$  column vector of encoded coefficients that capture all information at sensor  $s$ .

For CSSpf, LCpf, LCpf-GS and LApf, the joint log-likelihoods are computed as follows:

$$\alpha_s = (\Psi^T \Psi)^{-1} \Psi^T \gamma_s \quad (21)$$

$$\alpha = \sum_{s=1}^S \alpha_s = \sum_{s=1}^S (\Psi^T \Psi)^{-1} \Psi^T \gamma_s \quad (22)$$

$$\gamma = \Psi \alpha = \Psi \sum_{s=1}^S (\Psi^T \Psi)^{-1} \Psi^T \gamma_s \quad (23)$$

For Clusterpf, the vector  $\gamma_s$  instead denotes the  $C \times 1$  column vector of cluster log-likelihoods at sensor  $s$ . After the joint cluster log-likelihoods  $\gamma$  are computed, individual particle log-likelihoods are recovered via convex optimization. This last step renders the analysis of Clusterpf more complicated and we omit all discussions related to Clusterpf for now.



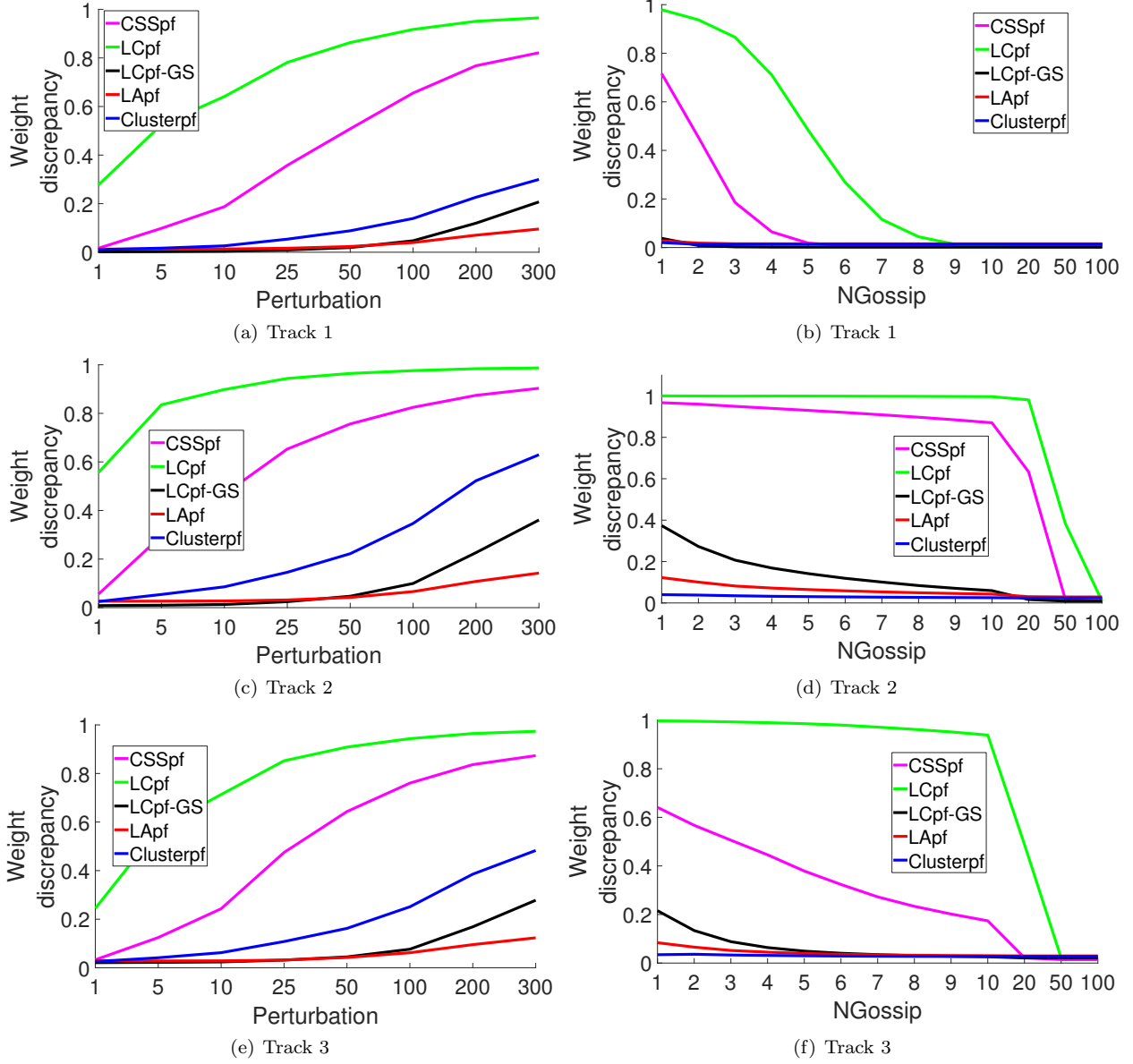


Figure 6: Average weight discrepancy between the exact particle weights (BSpf) and approximate particle weights (LCpf, LApf, Clusterpf) with respect to perturbation and NGossip. A perturbation of  $x$  represents  $x\%$  shift from the true aggregate value.  $N = 500$ ,  $d = 1$ ,  $m = C = 6$

With low communication overhead, the gossip algorithm does not converge and all sensors obtain  $\alpha' = \alpha + \Delta$ . This yields

$$\gamma' = \Psi\alpha' = \Psi\alpha + \Psi\Delta \quad (24)$$

where the term  $\Psi\Delta$  induces error in the joint log-likelihoods. Taking the exponential yields

$$w = \exp(\gamma') = \exp(\Psi\alpha) \exp(\Psi\Delta) \quad (25)$$

where the term  $\exp(\Psi\Delta)$  is the distortion induced on (unnormalized) particle weights.

In the ideal case, the gossip algorithm converges,  $\Delta = 0$ , and we have  $\exp(\Psi\Delta) = \mathbf{1}$  where  $\mathbf{1}$  is the vector of all 1's. In our perturbation test (Fig. 6), we compute and report  $\frac{1}{N} \|\mathbf{1} - \Psi\Delta\|_1$ . For each filter, we compute the  $N \times 1$  vector  $\mathbf{1} - \Psi\Delta$  and report its average over  $N$  particles. A value closer to 0 indicates less distortion to the particle weights due to gossiping error. Conversely, a value close to 1 indicates high distortion. Fig. 7 shows the results for all three tracks. In all perturbation tests, LAPf has the lowest curve, followed by LCPf-GS and Clusterpf. LCPf has the highest curve for all perturbation levels. On the other hand, in the NGossip tests, Clusterpf and LAPf have very similar level distortions. These trends are consistent with Fig. 6 which shows the discrepancy in normalized particle weights estimation. These results are also consistent with our previous observation that LAPf and Clusterpf have robust performance even at low communication overhead.

Our objective is to derive some suitable bounds on the distortion  $\|\Psi\Delta\|$ . From the submultiplicative property of  $L_2$  norm, we have

$$\|\Psi\Delta\| \leq \|\Psi\| \cdot \|\Delta\| \quad (26)$$

For LCPf-GS and LAPf, the encoding matrices are orthonormal and thus have norm 1. On the other hand,  $\|\Psi_{CSS}\|$  and  $\|\Psi_{LC}\|$  depend strongly on the states of particles  $x_i$  and the choice of basis functions.

We seek to bound the gossip error  $\|\Delta\|$ . Let  $W$  denote the mixing matrix in the gossip algorithm. Let  $k$  denote the current gossip iteration. We abuse slightly our existing notation and let  $\alpha_s(k)$  denote the coefficient values at sensor  $s$  after  $k$  gossip iterations. Each sensor starts with local coefficients  $\alpha_s(0)$ . The coefficients evolve as

$$\begin{bmatrix} \alpha_1(k+1)^T \\ \dots \\ \alpha_S(k+1)^T \end{bmatrix} = W \begin{bmatrix} \alpha_1(k)^T \\ \dots \\ \alpha_S(k)^T \end{bmatrix} \quad (27)$$

$$= W^{k+1} \begin{bmatrix} \alpha_1(0)^T \\ \dots \\ \alpha_S(0)^T \end{bmatrix} \quad (28)$$

$$= W^{k+1} \underbrace{\begin{bmatrix} \gamma_1^T \\ \dots \\ \gamma_S^T \end{bmatrix}}_{\text{Pseudo log-likelihood}} \Psi(\Psi^T \Psi)^{-1} \quad (29)$$

Eq. (29) suggests that, for all three filters, the highlighted terms may be interpreted as some type of pseudo log-likelihood. This allows us to readily apply existing results on gossiping error (on these pseudo log-likelihoods).

After  $k$  gossip iterations, max consensus is applied so all sensors obtain the same coefficients. This step is necessary to ensure that all sensors obtain the same particle weights and that the particle cloud remains synchronized across the network.

$$\gamma^T = \alpha^T \Psi^T = \text{MaxConsensus} \left( W^{k+1} \begin{bmatrix} \gamma_1^T \\ \dots \\ \gamma_S^T \end{bmatrix} \Psi(\Psi^T \Psi)^{-1} \right) \Psi^T \neq \text{MaxConsensus} \left( W^{k+1} \begin{bmatrix} \gamma_1^T \\ \dots \\ \gamma_S^T \end{bmatrix} \Psi(\Psi^T \Psi)^{-1} \Psi^T \right) \quad (30)$$

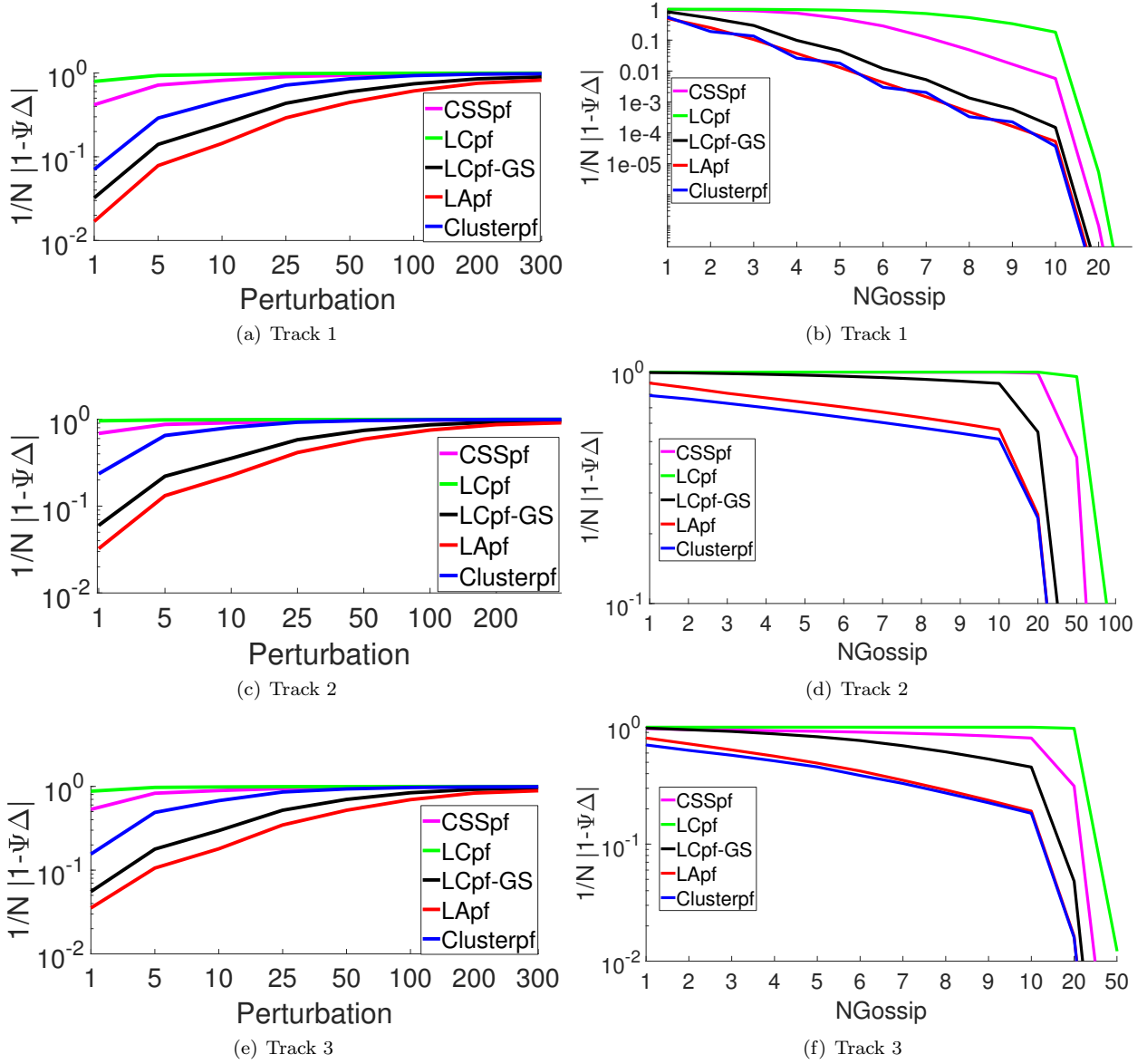


Figure 7: Average value of  $1 - \Psi\Delta$  over  $N$  particles. A perturbation of  $x$  represents  $x\%$  shift from the true aggregate value.  $N = 500$ ,  $d = 1$ ,  $m = C = 6$

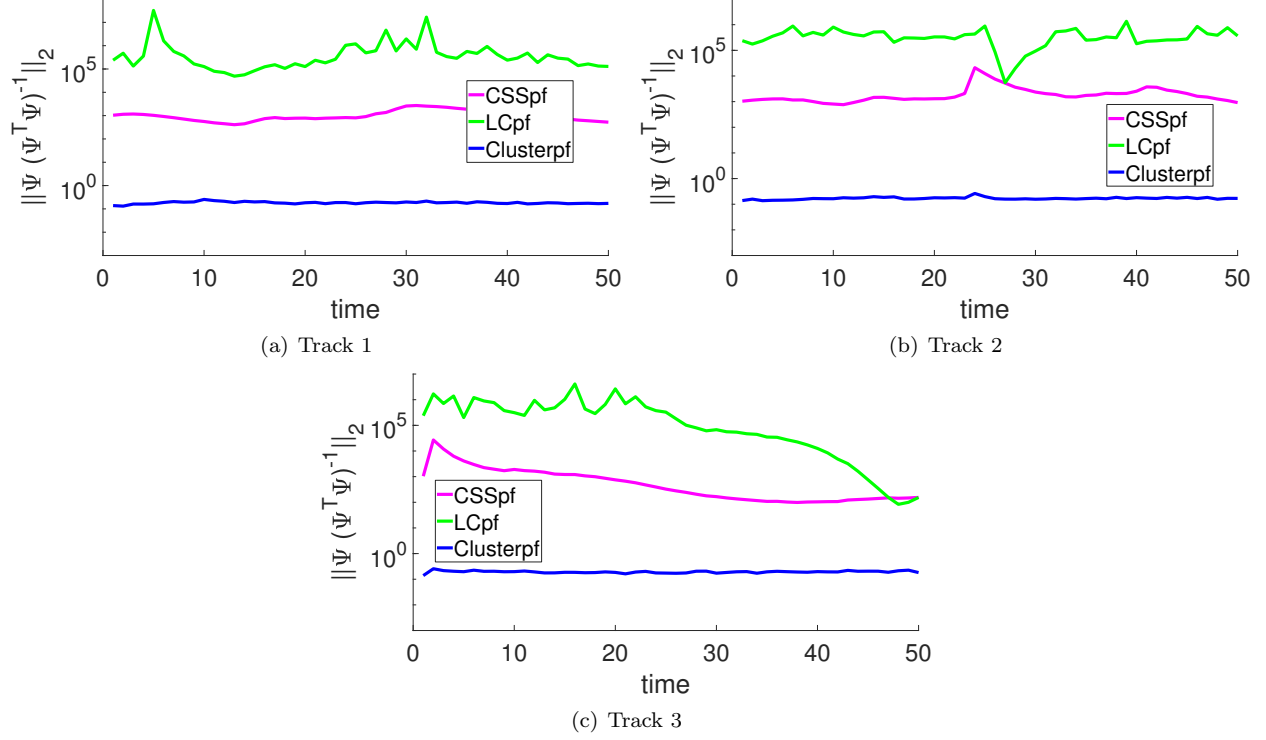


Figure 8: Average value of  $\|\Psi(\Psi^T \Psi)^{-1}\|_2$  over time for CSSpf, LCpf and Clusterpf

Fig. 8 shows the average norm of  $\|\Psi(\Psi^T \Psi)^{-1}\|_2$  over time for all three tracks. As expected, LCpf has the highest curve by several orders of magnitude; followed by CSSpf and Clusterpf. We omit the curves for LAPf and LCpf-GS since  $\|\Psi\|_2 = 1$  for these two filters.

In this set of tests, the tracking is done using a centralized BSpf. Thus, at each time step, the particles remain the same for all levels of perturbation and NGossip. In other words, the perturbation level and NGossip only affects  $W^{k+1} \begin{bmatrix} \gamma_1^T \\ \dots \\ \gamma_S^T \end{bmatrix}$  but has no impact on  $\|\Psi\|$ .

## 6 Conclusion

In this report, we present four distributed particle filters for single-target bearing-only tracking. CSSpf approximates the log-likelihood function using six sufficient statistics. LCpf uses likelihood consensus to encode the particle log-likelihoods. LAPf constructs a graph over all particles and uses the eigenvectors of resulting Laplacian matrix to encode the particle log-likelihood. Finally, Clusterpf groups particles into clusters, computes the cluster joint log-likelihood and recovers individual particle weights using convex minimization.

We study each individual algorithm's performance and compare them against each other. LCpf is very fast, but is highly susceptible to gossiping error and requires a much higher communication overhead to achieve adequate tracking performance. The LAPf and Clusterpf yield robust tracking performance in all tested scenarios; but they also have the highest runtime by a large margin. More interestingly, when minimal communication overhead is permitted, LAPf and Clusterpf are still able to yield robust performance while the other filters would break down. These results make LAPf and Clusterpf extremely attractive in energy-constrained tracking scenarios.

## References

- [1] A. Mohammadi and A. Asif, “A constraint sufficient statistics based distributed particle filter for bearing only tracking,” in *IEEE Int. Conf. Communications (ICC)*, Ottawa, ON, Canada, Jun 2012, pp. 3670–3675.
- [2] O. Hlinka, O. Sluciak, F. Hlawatsch, P. Djuric, and M. Rupp, “Likelihood consensus and its application to distributed particle filtering,” *IEEE Trans. Signal Process.*, vol. 60, no. 8, pp. 4334–4349, 2012.
- [3] M. Rabbat, M. Coates, and S. Blouin, “Graph laplacian distributed particle filtering,” in *Signal Process. Conf. (EUPISCO)*, Budapest, Hungary, Aug. 2016, pp. 1493 – 1497.
- [4] C. W. Chao, M. Rabbat, and S. Blouin, “Particle weight approximation with clustering for gossip-based distributed particle filters,” in *IEEE Int. Workshop Comp Comput. Advances Multi-Sensor Adaptive Process. (CAMSAP)*, Cancun, Mexico, Dec 2015, pp. 85–88.