Distributed Graph Laplacian Particle Filter

Jun Ye Yu, Michael Rabbat

May 3, 2018

Abstract

A key challenge in designing distributed particle filter is to minimize the communication overhead without compromising tracking performance. In this paper we present two distributed particle filters that achieve robust performance with low communication overhead. The two filters construct a graph of the particles and exploit the graph Laplacian matrix in different manners to encode the particle log-likelihoods using a minimum number of coefficients. We validate their performance via simulations with low overhead (i.e., less than 10 gossiping iterations) and provide a theoretical error bound for the presented filters.

1 Introduction

Particle filters are an effective solution for tracking targets with non-linear dynamic and/or measurement models. In a distributed setting, a network of sensors collect data and collaborate with each other to achieve improved tracking performance. These collaborations may involve dissemination of sensor measurements [1, 2], distributed computation of joint log-likelihoods [3–6] or propagation of posterior target distribution across sensors [7], etc., and may require considerable communication overhead (i.e., ensuring the data reaches the furthest sensor, reaching consensus on some global values, etc.). Since the sensors are often battery-powered, a key objective in designing a *distributed particle filter* (DPF) is to minimize the communication overhead without major degradation of the tracking performance.

There have been a large number of work on DPF in the past (see [8] for a survey). In this paper, we focus on consensus-based algorithms. Each sensor maintains a local particle-representation of target distribution and communicates only with its neighbors (i.e., no multi-hop transmissions). The objective is the computation of posterior target state distribution by incorporating data from all relevant sensors in the network. A naive and simple approach is to run one gossiping algorithm per particle to compute their joint log-likelihoods at the cost of prohibitively high communication overhead. One way to reduce the overhead amounts to reducing the number of particles. [9] uses min-consensus and max-consensus algorithms to determine a region of the target state space containing the particles with highest weights. This region is in turn used to construct an adapted proposal distribution so that fewer particles are required for actual tracking. [10] proposes an auxiliary particle filter that uses selective gossip so only particles with highest weights are communicated between sensors. Both methods improve over the naive approach, but may still incur a high communication overhead if the likelihood distribution is not peaky and/or the number of particles is high. An alternate approach is to model the local posterior distribution as either a Gaussian distribution [11] as or a mixture of Gaussians [12]. All sensors then broadcast the Gaussian parameters instead of individual particle weights. The local distributions can be fused via some fusion rule and the target state estimate is computed based on the global Gaussian (mixture) distribution. This approach may have poor performance if a Gaussian (mixture) is a poor fit for the true target distribution. A third approach is the distributed computation of joint particle log-likelihoods. In the likelihood consensus method [3], the log-likelihood function is approximated as a linear combination of basis functions. These basis functions are assumed known to all sensors and sensor-independent. Each sensor computes its local coefficients encompassing all sensor-dependent data. The joint log-likelihood function can then be recovered by aggregating all coefficients across the network via gossiping. This approach can achieve significant reduction in communication overhead when the approximation only requires a small number of coefficients (i.e., low-order polynomial functions). [13] and [6] are two special adaptations of likelihood consensus method in which the basis functions are specifically tuned for bearing-only tracking with additive Gaussian measurement noise.

In this paper we present two distributed particle filters based on graph Laplacian. The first method, the Laplacian particle filter, constructs a graph using the particles as vertices and uses the resulting Laplacian matrix to encode the particle log-likelihoods. The joint log-likelihoods are then recovered by aggregating the encoded coefficients across the network. The second method, the cluster particle filter, groups all particles into clusters, computes the joint cluster log-likelihoods and recovers the individual log-likelihoods via convex optimization involving graph Laplacian. Preliminary results of the two filters have been presented in [5] and [13]. In this paper, we present both filters in more details, provide more comprehensive simulation results and show that the proposed filters are able to yield robust tracking performance even with very low communication overhead (i.e., one gossip iteration per time step). We also derive a theoretical error bound for the proposed filters to provide insights into the robustness of said filters at low overhead.

The rest of the paper is organized as follows. Section 2 describes the tracking problem and provides a brief overview of distributed particle filtering. Section 3 describes the proposed filters. Simulation results are provided in Section 4 and theoretical analysis are provided in Section 5. Finally, Section 6 concludes the paper.

2 Problem formulation

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

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

$$X(k+1) = f(X(k), \xi(k)) \tag{1}$$

where f is the dynamic model and $\xi(k)$ is the process noise inducing the transition density $p(X(k+1)|X(k))$.

At every time step, each sensor s receives a measurement:

$$z^s(k) = h^s(X(k)), \eta(k)) \tag{2}$$

where h is the (sensor-dependent) measurement model and $\eta(k)$ is the measurement noise which induces the likelihood function $p(z^s(k)|X(k))$.

Let $Z(k) = \{z^1(k), ...z^S(k)\}$ denote the set of measurements from all sensors at time k and let $Z(1:k) = \{Z(1), ...Z(k)\}$. The objective is to estimate the posterior target distribution $p(X(k)|Z(1:k))$ based on the sequence of all available measurements up to time step k. If the distribution $p(X(k-1)|Z(1:k-1)$ is available, then $p(X(k)|Z(1:k)$ can be obtained recursively in two steps.

In the prediction step, we obtain a predicted density using the Chapman-Kolmogorov equation [14]:

$$p(X(k)|Z(1:k-1)) = \int p(X(k)|X(k-1))p(X(k-1)|Z(1:k-1)dX(k-1) \tag{3}$$

In the update stage where new measurements $Z(k)$ become available, the predicted density is updated as follows [14]:

$$p(X(k)|Z(1:k)) = \frac{p(Z(k)|X(k))p(X(k)|Z(1:k-1)}{p(Z(k)|Z(1:k-1))} \tag{4}$$

Eq. (3) and (4) form the basis of an optimal Bayesian solution; but they are in general computationally intractable. The particle filter provides a feasible approximation by modeling the posterior distribution using a set of N weighted particles: $\{X_i(k), w_i(k)\}_{i=1}^N$. More specifically, we have

$$p(X|Z(1:k)) \approx \sum_{i=1}^{N} w_i(k)\delta_{X_i(k)}(X) \tag{5}$$

where $\delta()$ is the Dirac Delta function and $\sum_i w_i(k) = 1$.

Let the weighted particles $\{X_i(k-1), w_i(k-1)\}_{i=1}^N$ represent the posterior distribution at time $k-1$. We first propagate the existing particles using a proposal distribution $q(X_i(k)|X_i(k-1), Z(1:k))$, and then compute the updated particle weights given new measurements $Z(k)$:

$$w_i(|k) \propto w_i(k-1)\frac{p(Z(k)|X_i(k))p(X_i(k)|X_i(k-1))}{q(X_i(k)|X_i(k-1), Z(1:k))} \tag{6}$$

If the transition density $p(X_i(k)|X_i(k-1))$ is used as the the proposal density, then the weight computation simplifies as $w_i(k) \propto w_i(k-1)p(Z(k)|X_i(k))$. Given the updated particle set $\{X_i(k), w_i(k)\}_{i=1}^N$, the *minimum-mean-squared-error* (MMSE) estimate can be computed as

$$\hat{X}(k) = \sum_{i=1}^{N} w_i(k)X_i(k) \tag{7}$$

Consider now a distributed implementation of the particle filter. We assume that the random number generators at all sensors are synchronized so all sensors maintain an identical set of particles and that all sensors agree on the particle weights $w_i(k-1)$. Computing the particle weights $w_i(k)$ therefore reduces to computing the joint likelihood $p(Z(k)|X_i(k))$. Under the standard assumption that measurements at different sensors are conditionally independent, the joint log-likelihoods factorize as follows:

$$\log(p(Z(k)|X_i(k))) = \sum_{s=1}^{S} \log(p(z^s(k)|X_i(k))) \tag{8}$$

Therefore, each sensor can compute $\log(p(z^s(k)|X_i(k))$ locally and obtain the sum using gossip algorithms [15, 16] or other consensus algorithms [17]. Since the number of particles can be quite high, running one gossip algorithm per particle would incur very high communication overhead. In the next section, we present two particle filters that compute the joint log-likelihoods in a distributed fashion with low communication overhead. For convenience of notation, we omit the time index k where there is no ambiguity.

3 Algorithms

3.1 Laplacian particle filter

The *Laplacian particle filter* (LApf) follows the likelihood consensus approach and models the log-likelihood function as a linear combination of m basis functions.

$$\log(p(z^s|X_i) \approx \sum_{j=1}^{m} \beta_j(X_i)\alpha_j^s \tag{9}$$

The basis functions $\beta_j(X_i)$ depend only on the particles X_i and are known to all sensors. In contrast, the coefficients α_j^s are computed locally at each sensor. The joint log-likelihood can be computed as follows:

$$\gamma(X_i) = \log(p(Z|X_i) \approx \sum_{s=1}^{S}\sum_{j=1}^{m} \beta_j(X_i)\alpha_j^s = \sum_{j=1}^{m} \beta_j(X_i)\left(\sum_{s=1}^{S}\alpha_j^s\right) \tag{10}$$

In other words, since $\beta_j(X_i)$ are known to all sensors and can be computed locally, it suffices to compute the m aggregate coefficients to recover the joint log-likelihoods. When $m \ll N$, the communication overhead can be reduced significantly. In the likelihood consensus approach [3], a low-order polynomial expansion is used as the basis functions. In LApf, we choose to use a set of functions more adapted to the particle filters' positions.

We consider each particle X_i as a vertex on a graph and use Delaunay triangulation to generate the edges. Let A denote the $N \times N$ adjacency matrix where $A(i,j) = A(j,i) = 1$ if X_i and X_j are connected. Let D denote the $N \times N$ matrix where $D(i,i) = \sum_{j=1}^{N} A(i,j)$ and $D(i,j) = 0 \quad i \neq j$. Let $L = D - A$ denote the Laplacian matrix of the particle graph. Since L is a real symmetric matrix, it has the eigendecomposition $L = \Psi\Lambda\Psi^T$ where Λ is a diagonal matrix of eigenvalues with corresponding eigenvectors as columns of Ψ and the superscript T denotes the transpose operation. These eigenvectors can be used as a fourier basis [18] for the signals supported on the graph. The log-likelihood vector $\gamma^s = [\gamma^s(X_1), ...\gamma^s(X_N)]^T$ is one such signal and can be encoded as follows:

$$\alpha^s = \Psi_m^T \gamma^s \tag{11}$$

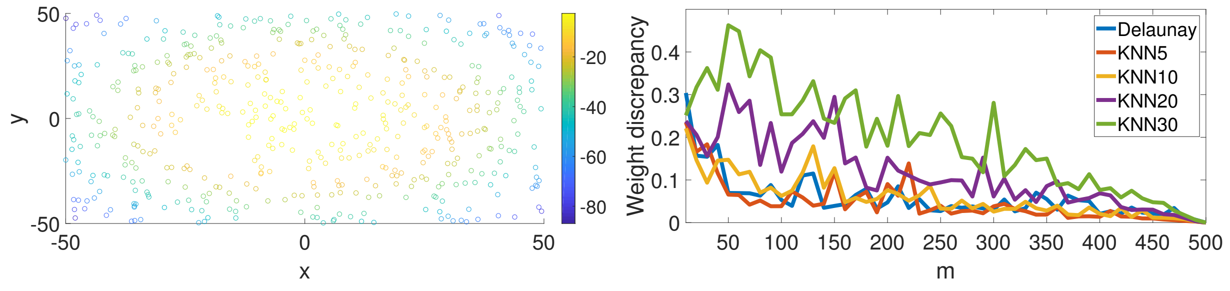
where Ψ_m is a matrix consisting of m column eigenvectors. The log-likelihood can then be recovered as follows:

$$\hat{\gamma}^s = \Psi_m \alpha^s = \Psi_m \Psi_m^T \gamma^s \tag{12}$$

In the multi-sensor setting, m gossiping algorithms are run in parallel to compute the coefficients $\alpha_j = \sum_s \alpha_j^s$. The number of coefficients m can either be selected beforehand and set by individual sensors to ensure the discrepancy $\sum_{i=1}^{N} |\gamma^s(X_i) - \gamma(X_i)|$ (or other suitable distance metric) is sufficiently low. In the latter case, a max-consensus algorithm can be used to determine the maximum value of m among all sensors.

The final question remains as to the choice of the m eigenvectors. 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 now derive the overhead of computing joint log-likelihoods under LApf. The Delaunay triangulation for graph construction has complexity $O(N)$ [19]. The eigenvalue decomposition has complexity $O(N^3)$ [20]. Computing the m coefficients and recovering the joint log-likelihoods both have complexity $O(mN)$. During each gossip iteration, m scalars are broadcast by all sensors. The overall complexity is thus $O(SN + SN^3 + SmN + SmNGossip + SmN) \subset O(SN^3)$ where $NGossip$ is the number of gossiping iterations. For comparison, the likelihood consensus does not require the graph construction nor the eigenvalue decomposition and thus has overhead $O(SmNGossip + SmN)$. For $m \ll N$ and $NGossip \ll N$, the likelihood consensus has much lower overhead; but we will show that LApf yields robust performance even at low $NGossip$ whereas the likelihood



(a) Scatter plot of particles with colors corresponding to log-likelihoods

(b) Weight discrepancy with respect to m averaged over 200 random trials

Figure 1: Weight discrepancy of LApf with respect to m and different graph construction methods

consensus would break down. We will also show that using eigenvectors to encode the log-likelihoods has a direct impact on the robustness of the filter.

Fig. 1(a) shows a scatter plot of particle cloud with colors corresponding to their log-likelihoods. Note that the colors do exhibit a smooth transition. Fig. 1 (b) show the discrepancy $\sum_{i=1}^{N} |\gamma^s(X_i) - \gamma(X_i)|$ with respect to m averaged over 200 random trials. The target remains fixed but the particles are randomly placed in the 100×100 area at each trial. We consider both Delaunay triangulation and K-nearest-graph methods for graph generation. The Delaunay graph yields fairly low discrepancy without the need for tuning an additional parameter. For the remainder of our paper, we consider only Delaunay triangulation for graph construction in LApf.

3.2 Cluster particle filter

A key challenge of the Laplacian particle filter is the high computational burden of the eigenvalue decomposition. The *Cluster particle filter* (CLusterpf) seeks to exploit the graph Laplacian while avoiding the computational burden of LApf.

In Clusterpf, the particles are grouped into K clusters based on their position. Let C denote the $K \times N$ cluster assignment matrix where $C(i, j) = 1$ if particle j belongs to cluster i. The log-likelihood of the i^{th} cluster, γ_c^i, is equal to the aggregate log-likelihoods of its constituent particles. We can thus relate the cluster log-likelihoods to the particle log-likelihoods as follows:

$$\gamma_c = C\gamma \tag{13}$$

Rather than gossiping on the $N \times 1$ vector γ^s, each sensor only needs to broadcast the $K \times 1$ vector γ_c. When $K \ll N$, significant reduction in communication overhead can be achieved. After reaching consensus on γ_c, we need to recover the individual particle joint log-likelihoods. A naive solution is to simply assign equal weight to all particles in a cluster; but this leads to poor results as the resulting log-likelihoods exhibit sharp change at cluster boundary. Instead, we exploit the graph Laplacian to ensure that the log-likelihood values remain smooth over the state space. We again construct a Delaunay triangulation graph over the particles, compute the Laplacian matrix L, and then solve the following convex minimization problem:

$$\underset{\gamma}{\text{minimize}} \quad \gamma^T L \gamma \tag{14}$$

$$\text{subject to} \quad C\gamma = \gamma_c \tag{15}$$

In other words, we seek to assign particle log-likelihood values that are smooth with respect to particle proximity while ensuring the aggregate values are equal to the cluster value. Since the matrix L is positive semi-definite, the problem is convex, has a global minimum and can be solved using well-known methods [21].

We now derive the computational overhead of the cluster filter. K-means clustering has complexity $O(4NKI)$ [22] where I is the number of clustering iterations and the constant 4 is the target state dimension. K 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 $O(\sqrt{N})$ [23]. The overall complexity is thus $O(SNKI + SKNGossip + SN + S\sqrt{N}) \subset O(SNKI)$. For $K \ll N$ and $I \ll N$, Clusterpf has a significantly lower computational overhead than LApf.

Fig. 2 shows the average weight discrepancy of Clusterpf with respect to K. At low K, the Delaunay triangulation graph offers competitive performance without the need of tuning an additional parameter. For the remainder of the paper, we consider only Delaunay triangulation for graph construction in Clusterpf.

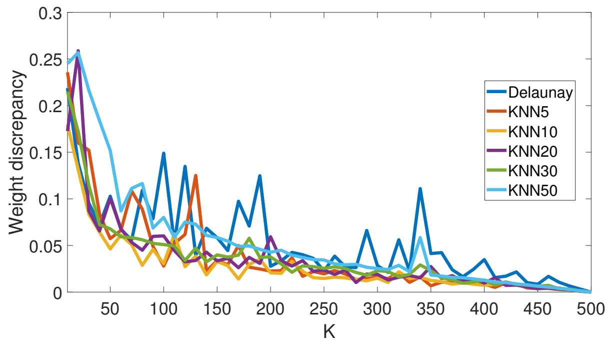
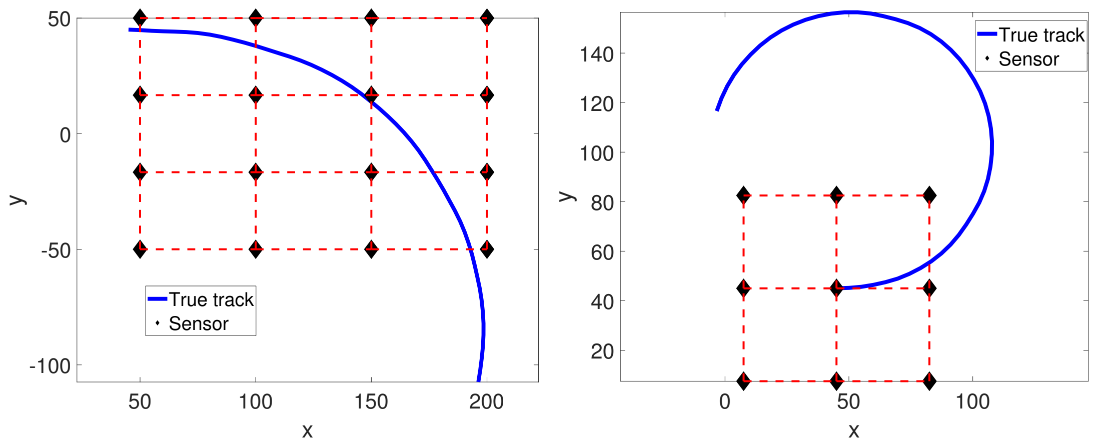


Figure 2: Weight discrepancy with respect to K averaged over 200 trials

4 Numerical study

In this section, we evaluate the performance of the two presented filters. For comparison, we also evaluate the *likelihood consensus particle filter* (LCpf) [3], *constraint sufficient statistics particle filter* (CSSpf) [4] and include a centralized bootstrap particle filter as baseline. We consider two different simulated tracks with different measurement models. Each track contains a network of S static sensors which collaboratively track a moving target over 50 time steps. Fig. 3 shows the two test tracks.



(a) Track 1 (b) Track 2

Figure 3: Target tracks (blue curve) and sensor positions (black diamond). Sensors connected by red dashed lines are within broadcast range of each other.

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

$$X(k+1) = F(X(k)) + \xi(k) \tag{16}$$

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

$$F(X(k)) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

For coordinated turn, we have

$$F(X(k)) = \begin{bmatrix} 1 & 0 & \frac{\sin(\Omega)}{\Omega(k)} & -\frac{1-\cos(\Omega(k))}{\Omega(k)} \\ 0 & 1 & \frac{1-\cos(\Omega(k))}{\Omega(k)} & \frac{\sin(\Omega(k))}{\Omega(k)} \\ 0 & 0 & \cos(\Omega(k)) & -\sin(\Omega(k)) \\ 0 & 0 & \sin(\Omega(k)) & \cos(\Omega(k)) \end{bmatrix} \tag{18}$$

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

$$\Omega(k) = \frac{a}{\sqrt{\dot{x}^2(k) + \dot{y}^2(k)}} \tag{19}$$

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

In the first simulated track, all sensors receive noisy bearing measurements (in radians) from the target.

$$H_s(X(k)) = \arctan 2\left(\frac{x_t(k) - x_s}{y_t(k) - y_s}\right) + \eta(k) \tag{20}$$

where $\eta(k)$ is the zero-mean Gaussian measurement noise. In the second track, the sensors receive noisy range measurements (in km).

$$H_s(X(k)) = \sqrt{(x_t(k) - x_s)^2 + (y_t(k) - y_s)^2} + \eta(k) \tag{21}$$

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

$$Q = \sigma_a^2 \begin{bmatrix} \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 1 & 0 \\ 0 & \frac{1}{2} & 0 & 1 \end{bmatrix} \tag{22}$$

$$R_{\text{bearing}} = \sigma_\theta^2 \tag{23}$$

$$R_{\text{range}} = \sigma_r^2 \tag{24}$$

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

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 $0 \le i, j \le d$ where d is some user-specified max polynomial degree (i.e., For $d = 2$, the basis functions would be $\beta_1(X) = x_t^0 y_t^0 = 1, \beta_2(X) = x_t^0 y_t^1 = y_t, ..., \beta_9(X) = x_t^2 y_t^2.$). We also include a modified version of LCpf. We define the matrix Ψ^{LC} where $\Psi^{LC}(i,j) = \beta_j(X_i)$. We apply Gram-Schmidt orthogonalization to Ψ^{LC} so that all columns are orthogonal to each other and have unit norm. The columns of Ψ^{LC} thus provide a basis for encoding log-likelihoods and have a role analogous to the eigenvectors in LApf. This filter is henceforth referred to as LCpf-GS and the reason for its inclusion will be explained in the next section. For LApf, we construct a Delaunay triangulation graph and retain $m < N$ eigenvectors as the basis of Laplacian transformation. Finally, for Clusterpf, all particles are grouped into K clusters and a Delaunay triangulation graph is constructed to recover individual particle weights. Unless otherwise specified, we set $d = 2$ so LCpf has 9 basis functions, and we set $m = K = 6$. These values are chosen to match CSSpf which only uses 6 coefficients to encode the log-likelihoods and to minimize the communication overhead per sensor per gossip iteration.

The random number generators are synchronized to ensure that the particles remain the same across sensors. Distributed summation is performed using gossip algorithm. 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:

$$G_{i,\text{new}} = w_{ii} G_{i,\text{old}} + \sum_{j \in N_i} w_{ij} G_{j,\text{old}} \tag{25}$$

$$w_{ij} = \begin{cases} \frac{1}{1+max(d_i,d_j)} & j \in N_i \\ 1 - \sum_{j \in N_i} w_{ij} & i = j \\ 0 \quad j \notin N_i \end{cases} \tag{26}$$

where N_i denotes the set of neighboring sensors of sensor i, $d_i = |N_i|$, and Metropolis weight is used for the update. Since only a finite number of gossiping iterations are executed, all sensors are not guaranteed to obtain the same values. Therefore, a max-consensus algorithm is executed to ensure all sensors obtain the same values. 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 $G_{\text{initial}} = [G_1, ..., G_S]^T$, the final values can be easily computed as

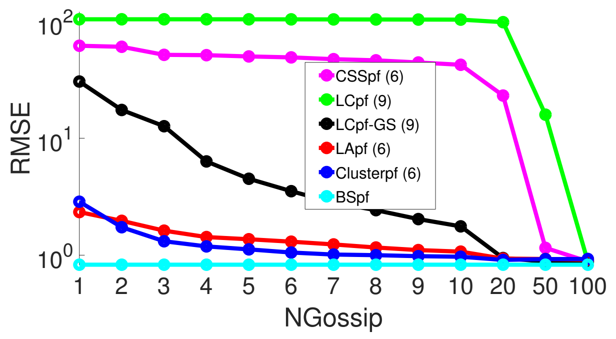
$$G_{\text{final}} = W^{NGossip} G_{\text{initial}} \tag{27}$$

In the remainder of the section, we run a number of Monte Carlo simulations to compare the performance of all filters. The track and the sensor positions remain 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 total runtime. The first metric is self-explanatory and the second metric compares the computational overhead of all filters. Note that CSSpf is not run for track 2 since it is designed for bearing-only tracking.

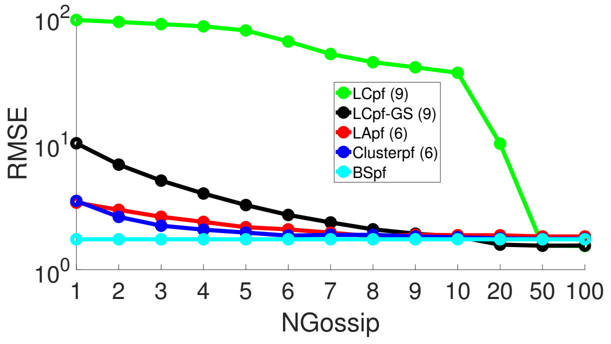
Fig. 4 shows the simulation results for both tracks. Consider track 1 first (left column figures). For $NGossip \le 100$, CSSpf and LCpf break down and have very high RMSE. After crossing a threshold value (50 for CSSpf and

100 for LCpf), the RMSE drops sharply. At low $NGossip$, LApf and Clusterpf outperform the other distributed filters by a significant margin and are on-par with the centralized BSpf for $NGossip \geq 20$. More interestingly, these two filters have robust tracking performance even with just 1 gossip iteration per time step. LCpf-GS has better performance than CSSpf and LCpf but still breaks down at low $NGossip$. The LApf has the highest runtime by a significant margin due to eigenvalue decomposition as expected. The clusterpf has the second highest runtime which can be attributed to the K-means clustering and convex minimization. The LCpf-GS and CSSpf have the lowest runtime. Note that the total runtime is fairly constant over all values of $NGossip$ due to our implementation of the gossiping algorithm.

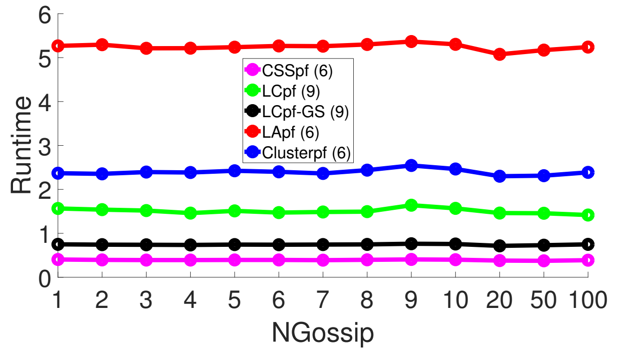
Consider the results for track 2 (right column figures). The RMSE curves display the same trends as in track 1. The LCpf breaks down at low $NGossip$ and its RMSE drops sharply for $NGossip > 10$. The LApf and Clusterpf consistently outperform the other distributed filters at low overhead. The LCpf-GS again outperforms LCpf and achieves performance on-par with LApf for $NGossip \geq 8$. The curves for total runtime exhibit the same trends as in track 1 with LApf being the slowest by far.



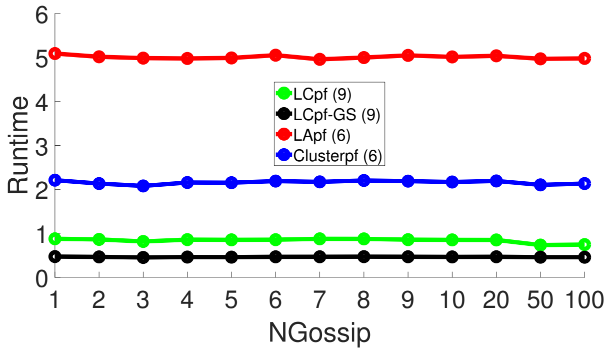
(a) Track 1, RMSE



(b) Track 2, RMSE



(c) Track 1, Total runtime



(d) Track 2, Total runtime

Figure 4: RMSE and total runtime, averaged over 200 Monte Carlo trials for test tracks 1 and 2. The number in brackets indicate the number of scalars broadcast per sensor per gossip iteration.

5 Theoretical analysis

Let $\hat{\gamma}(X_i)$ denote the approximate joint particle log-likelihood for particle X_i and let $\gamma(X_i)$ denote the exact value. In this section, our objective is to derive an upper bound for $\frac{||\hat{\gamma}(X_i)-\gamma(X_i)||}{||\gamma(X_i)||}$ and to understand the robustness of the proposed filters even at low gossiping iterations.

All distributed filters presented in the simulations follow these same general steps:

1. Each sensor computes local particle log-likelihoods.
2. Each sensor encodes the local log-likelihoods into m coefficients.
3. Gossip and max-consensus algorithms are used to compute the m aggregate coefficients.
4. Approximate joint log-likelihoods are recovered from the aggregate coefficients.

Therefore, the discrepancy between true log-likelihoods $\gamma(X_i)$ and the final approximate value $\hat{\gamma}(X_i)$ comes from two sources: the encoding error and the gossiping error. The first error is the discrepancy between true log-likelihoods and their reconstructions from m coefficients. The second error occurs when sensors do not obtain the true aggregate coefficient values after only a finite number of gossiping iterations.

Consider the encoding error first. Let $\gamma_m(X_i)$ denote the approximate log-likelihood recovered from m coefficients. We assume that the following bound on encoding error holds for all particles

$$|\gamma_m(X_i) - \gamma(X_i)| \leq \delta_m |\gamma(X_i)| \tag{28}$$

The ratio term $0 \leq \delta_m \leq 1$ depends on the specific filter and the number of coefficients m. Given a required threshold δ_m, each sensor can choose a suitable value of m; then max-consensus algorithm can be used to agree on a global value. Note that Eq. (28) can be rewritten to yield

$$(1 - \delta_m)|\gamma(X_i)| \leq |\gamma_m(X_i)| \leq (1 + \delta_m)|\gamma(X_i)| \tag{29}$$

Consider the gossiping error. Let $\hat{\alpha}$ denote the coefficients obtained after gossiping and max-consensus iterations. This gives us the approximate log-likelihoods $\hat{\gamma}$. We assume that the following error ratio holds for all particles

$$\frac{|\gamma_m(X_i) - \hat{\gamma}(X_i)|}{|\gamma_m(X_i)|} \leq \delta_{Gossip} \tag{30}$$

Putting everything together, we obtain

$$\begin{align}
\frac{|\hat{\gamma}(X_i) - \gamma(X_i)|}{|\gamma(X_i)|} &= \frac{|\hat{\gamma}(X_i) - \gamma_m(X_i) + \gamma_m(X_i) - \gamma(X_i)|}{|\gamma(X_i)|} \tag{31}\\
&\leq \frac{|\hat{\gamma}(X_i) - \gamma_m(X_i)|}{|\gamma(X_i)|} + \frac{|\gamma_m(X_i) - \gamma(X_i)|}{|\gamma(X_i)|} \tag{32}\\
&\leq (1 + \delta_m)\frac{|\hat{\gamma}(X_i) - \gamma_m(X_i)|}{|\gamma_m(X_i)|} + \frac{|\gamma_m(X_i) - \gamma(X_i)|}{|\gamma(X_i)|} \tag{33}\\
&\leq (1 + \delta_m)\delta_{gossip} + \delta_m = \delta \tag{34}
\end{align}$$

Consider Eq. (34). When both the encoding and gossiping errors are present, they have a cascading effect on the error propagation. The overall error is bounded (i.e., less than unity) when both terms are bounded and less than unity. In fact, $\delta \leq 1$ is one of the conditions required to establish a time-uniform bound on the overall error of particle filters [24]. The other conditions are

1. The Markov chain associated with target state transition undergoes sufficient mixing within a finite number of time steps.
2. The state estimate function $F(X_i)$ can be suitably scaled and bounded such that $\sup_{X_i} |F(X_i)| < 1$
3. The particle likelihood is bounded such that $\exp(\gamma(X_i)) \leq 1 \quad \forall X_i$.

Note that fulfillment of the the first two conditions is unrelated to the choice of the filter. The third condition can be easily satisfied with suitable normalization of log-likelihoods. Thus our discussion focuses only on the error bound δ. More specifically, we focus on δ_{gossip} since δ_m can be reduced by simply increasing the number of coefficients.

Let $\hat{\alpha}(l)$ be the coefficients obtained after l gossip iterations and max-consensus iterations, and let α denote the true coefficient values. We seek to establish the following bound for all coefficients:

$$\frac{|\hat{\alpha}_j(l) - \alpha_j|}{|\alpha_j|} \leq \beta, \quad 1 \leq j \leq m \tag{35}$$

Let S denote the total number of sensors, let W denote the $S \times S$ averaging matrix used in the gossiping algorithm and let ρ_W denote its second largest eigenvalue in modulus. The bound in Eq. (35) holds if the minimum number of gossip iteration satisfies the following condition [25]

$$l \geq \frac{1.5\log(S) + \log(\frac{S-1}{\beta})}{\log(1/\rho_W)} \tag{36}$$

For CSSpf, LCpf, LCpf-GS and LApf, we can define a linear transformation between the coefficients α and the approximate log-likelihoods $\hat{\gamma}$. For CSSpf and LCpf which use m basis functions for encoding, we have the $N \times m$ matrix $\Psi^{LC/CSS}$ where $\Psi(i,j) = \beta_j(X_i)$. For LCpf-GS, we start with the same matrix Ψ^{LC} and apply Gram-Schmidt process to obtain an orthonormal matrix Ψ^{LC-GS}. For LApf, the matrix Ψ^{LA} simply contains the m eigenvectors corresponding to the m smallest eigenvalues of the graph Laplacian. Finally, for Clusterpf, the cluster assignment matrix C can be interpreted as a linear transformation from particle log-likelihoods to cluster log-likelihoods; but the reverse process is not a linear transformation. To summarize, we have

$$\begin{align}
\gamma_m^{\text{CSS}} &= \Psi^{\text{CSS}}\alpha^{\text{CSS}} \tag{37}\\
\gamma_m^{\text{LC}} &= \Psi^{\text{LC}}\alpha^{\text{LC}} \tag{38}\\
\gamma_m^{\text{LC-GS}} &= \Psi^{\text{LC-GS}}\alpha^{\text{LC-GS}} \tag{39}\\
\gamma_m^{\text{LA}} &= \Psi^{\text{LA}}\alpha^{\text{LA}} \tag{40}\\
\gamma_m^{\text{Cluster}} &= \arg\min_{\gamma} \gamma^T L \gamma \quad C\gamma = \alpha^{\text{LC}} \tag{41}
\end{align}$$

We can derive an upper bound for the gossiping error of CSSpf, LCpf, LCpf-GS and LApf as follows:

$$\begin{aligned}\frac{|\hat{\gamma}(X_i) - \gamma_m(X_i)|}{|\gamma_m(X_i)|} &\leq \frac{||\hat{\gamma} - \gamma||_\infty}{|\gamma_m(X_i)|} \\ &= \frac{||\Psi(\hat{\alpha} - \alpha)||_\infty}{|\gamma_m(x_i)|} \\ &\leq \frac{||\Psi||_\infty ||\hat{\alpha} - \alpha||_\infty}{(1-\delta_m)|\gamma(x_i)|} \\ &\leq \frac{\beta ||\Psi||_\infty ||\alpha||_\infty}{(1-\delta_m)\min_{x_i} |\gamma(x_i)|} \end{aligned} \tag{42}$$

where the third line follows from triangular inequality and Eq. (28) and the last line follows from Eq. (35).

To validate the derived error bounds, we run the following simulations. We run a centralized bootstrap filter. At each time step, for comparison, we re-compute the particle weights using all the distributed filters. We compute and report the following the values:

1. $\max_{X_i} \frac{|\gamma_m(X_i) - \gamma(X_i)|}{|\gamma(X_i)|}$: estimate of δ_m
2. $\max_{x_i} \frac{|\gamma_m(X_i) - \hat{\gamma}(X_i)|}{|\gamma_m(X_i)|}$: estimate of δ_{gossip}
3. $\max_{\alpha_j} \frac{|\hat{\alpha}_j - \alpha_j|}{|\alpha_j|}$: estimate of β
4. $\frac{\beta||\Psi||_\infty ||\alpha||_\infty}{\min_{x_j} |\gamma(x_j)(1-\delta_m)|}$: upper limit of δ_{gossip} for LApf, LCpf, LCpf-GS and CSSpf
5. $\max_{x_i} \frac{|\hat{\gamma}(x_i) - \gamma(x_i)|}{|\gamma(x_i)|}$: estimate of δ
6. $(1+\delta_m)\delta_{gossip} + \delta_m$: upper limit of δ
7. $||w_{true} - w_{approx}||_2$: discrepancy of normalized particle weights

Fig. 5 shows the results for track 1. All data points are averaged over 40 MC trials. Consider first the encoding error δ_m. The LCpf and LCpf-GS have the lowest encoding error followed by CSSpf. The LApf and Clusterpf have the highest encoding error. For all filters, the encoding error spikes for time steps 22 to 23. Otherwise, δ_m remains below 1. Note that, in our set-up, the actual tracking is done using a centralized BSpf. Therefore, $\gamma_m(X_i)$ and by extension δ_m remains constant over different values of NGossip.

Consider next the gossiping error. For Clusterpf, the gossiping error is consistently below unity. For LApf and LCpf-GS, the gossiping error drops below 1 for NGossip>4; although the error for both filters is fairly close to 1 even at low NGossip. For CSSpf and LCpf, δ_{gossip} falls below 1 only when NGossip exceeds 50 and 75. We also plot the derived upper bounds. While these bounds are quite loose, their overall trends are consistent with the actual values.

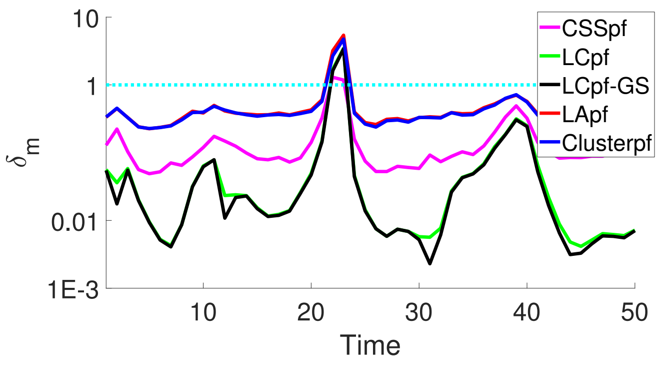
Fig. 5 (c) shows the average value of $\beta = \max_j |\hat{\alpha}_j - \alpha_j|/|\alpha_j|$. The β values for all filters are very close. This is not surprising since β is independent of the choice of the filters. Consider Eq. (42). In the ideal case of $\delta_m = 0$ (i.e., perfect reconstruction of log-likelihoods from m coefficients), the denominator of the bound is the same for all algorithms. From the numerator, we have $\delta_{\text{gossip}} \propto \beta||\Psi||_\infty ||\alpha||_\infty$ which makes intuitive sense. If LApf and LCpf are to achieve the same level of gossiping error, then we should have

$$\frac{\beta^{LC}}{\beta^{LA}} = \frac{||\Psi^{LA}||_\infty ||\alpha^{LA}||_\infty}{||\Psi^{LC}||_\infty ||\alpha^{LC}||_\infty} \tag{43}$$

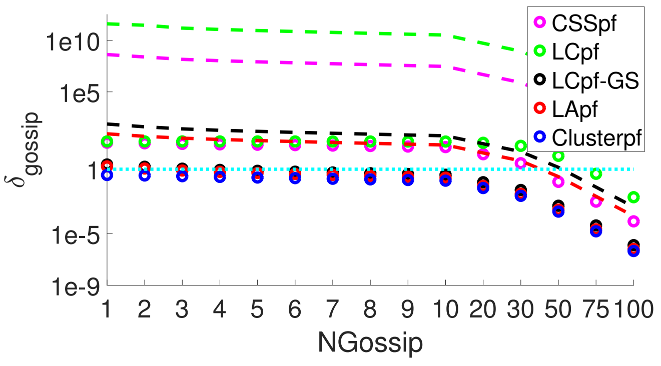
Fig. 5 (d) shows the average value of $||\Psi||_\infty ||\alpha||_\infty$ over time for CSSpf, LCpf, LCpf-GS, and LApf. The LCpf has the largest curve by several order of magnitude followed by CSSpf. Conversely, the LApf has the lowest curve followed by LCpf-GS. This, combined with Eq. (43) suggests that LCpf and CSSpf would indeed require significantly more gossiping iterations to reduce β accordingly. For LCpf-GS, the orthogonalization of the encoding matrix reduces the value of $||\Psi||_\infty ||\alpha||_\infty$. As a result, LCpf-GS requires less gossiping iterations than LCpf to achieve robust tracking performance. Conversely, the proposed LApf and CSSpf have low $||\Psi||_\infty ||\alpha||_\infty$ which allows for higher β (i.e., less gossiping iterations and more errors in coefficient values). We note that these analyses do not apply to Clusterpf since it does not have a linear transformation from coefficients to log-likelihoods. But empirically the Clusterpf does indeed have low gossiping error on-par with that of LApf.

Fig. 5 (e) shows the overall error δ. For Clusterpf, $\delta < 1$ for all values of NGossip. For LApf and LCpf-GS, $\delta < 1$ for $NGossip > 4$. For LCpf and CSSpf, the error becomes bounded only when NGossip exceeds 50 and 75 respectively. For all filters, the upper bound $(1+\delta_m)\delta_{\text{gossip}} + \delta_m$ is very close to the true value. As δ_m is small for all filters, at low NGossip, the gossiping error δ_{gossip} becomes the dominating factor and Clusterpf has the lowest δ. At higher NGossip, the encoding error becomes the dominating factor and LCpf-GS has the lowest δ.

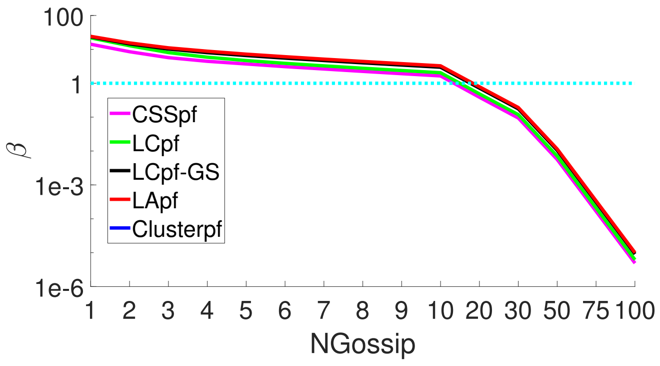
Finally, Fig. 5 (f) shows the average discrepancy in normalized particle weights. The curves for CSSpf and LCpf are significantly higher than the other curves until NGossip > 50 and NGossip > 75 while Clusterpf has the lowest curve as expected.



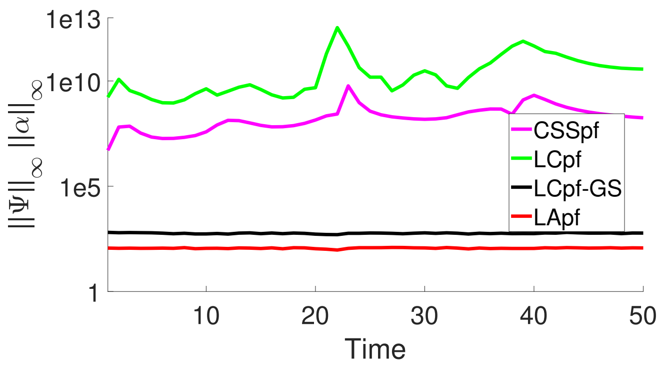
(a) Encoding error δ_m



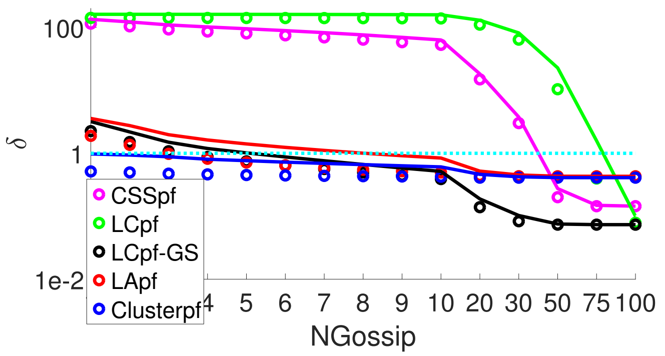
(b) Gossiping error δ_{gossip}, circles represent true gossiping errors and solid lines represent the theoretical upper bound.

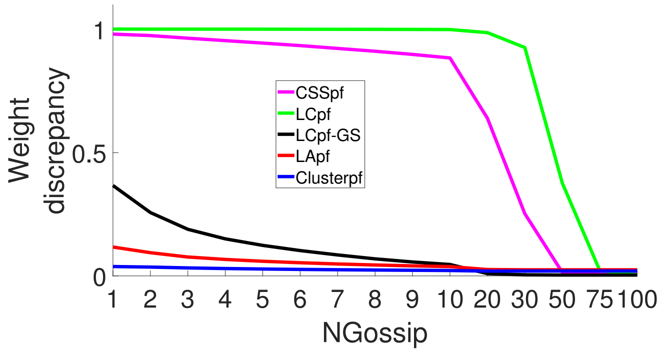


(c) β, the error ratio of coefficients α



(d) Average $||\Psi||_\infty ||\alpha||_\infty$ over time





(e) Log-likelihood error δ, circles represent true errors and dashed lines represent the theoretical higher bound $(1 + \delta_m)\delta_{\text{gossip}} + \delta_m$.

(f) Normalized particle weights discrepancy $||w_{\text{true}} - w_{\text{approx}}||_2$

Figure 5: Distributed particle filter error bounds with respect to *NGossip* and time for track 1. $N = 500$, $d = 2, m = C = 6$. The cyan dotted line denotes the $y = 1$ threshold value.

6 Conclusion

In this paper we present two distributed particle filters that achieve robust tracking performance with low communication overhead. Both filters construct a graph of the particles and exploit the graph Laplacian matrix to encode the particle log-likelihoods using a small number of coefficients. We validate their performance via simulations and derive a theoretical error bound that provides key insights into their robust performance.

References

[1] M. Rosencrantz, G. J. Gordon, and S. Thrun, "Decentralized sensor fusion with distributed particle filters," in *Procs. Uncertainty Artificial Intell. (UAI)*, no. 493-500, Acapulco, Mexico, Aug. 2003.

[2] M. Coates, "Distributed particle filters for sensor networks," *Procs. IPSN.*, pp. 99–107, Apr. 2004.

[3] 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.

[4] A. Mohammadi and A. Asif, "Distributed consensus + innovation particle filtering for bearing/range tracking with communication constraints," *IEEE Trans. Signal Process.*, vol. 63, pp. 620–635, Nov. 2014.

[5] M. Rabbat, M. Coates, and S. Blouin, "Graph laplacian distributed particle filtering," in *Signal Process. Conf. (EUPISCO)*, Budapest, Hungary, Aug. 2016, pp. 1493 – 1497.

[6] J. Y. Yu, M. Coates, M. Rabbat, and S. Blouin, "A distributed particle filter for bearings-only tracking on spherical surfaces," *IEEE Signal Process. Lett.*, vol. 23, pp. 326–330, Jan. 2016.

[7] Z. Yan, B. Zheng, and J. Cui, "Distributed particle filter for target tracking in wireless sensor network," in *Proc. EUSIPCO*, Sep. 2006.

[8] O. Hlinka, F. Hlawatsch, P. Djuric, and P. M. Djuric, "Distributed particle filtering in agent networks: A survey, classification, and comparison," *IEEE Signal Process. Mag.*, vol. 30, pp. 61–81, Dec. 2012.

[9] S. Farahmand, S. I. roumeliotis, and G. B. Giannakis, "Set-membership constrainted particle filter: distributed adaptation for sensor networks," *IEEE Trans. Signal Process.*, vol. 59, pp. 4122–4138, Jun. 2011.

[10] D. Ustebay, M. Coates, and M. Rabbat, "Distributed auxiliary particle filters using selective gossip," in *Procs. IEEE Int. Conf. Acoustics, Speech, Signal Process. (ICASSP)*, Prague, Czech Republic, May 2011, pp. 3296–3299.

[11] D. Gu, J. Sun, Z. Hu, and H. Li, "Consensus based distributed particle filter in sensor networks," in *Int. Conf. Inform. Automation*, Changsha, China, Aug. 2008, pp. 302–307.

[12] X. Sheng, Y. H. Hu, and P. Ramanathan, "Distributed particle filter with GMM approximation for multiple targets localization and tracking in wireless sensor network," in *4th Int. Symp. Inform. Process. Sensor Networks*, Boise, ID, USA, Apr. 2005, pp. 181–188.

[13] 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.

[14] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter.* Artech House, 2003.

[15] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans Inf. Theory*, pp. 2508–2530, Jun. 2006.

[16] D. Ustebay, R. Castro, and M. Rabbat, "Efficient decentralized approximation via selective gossip," *IEEE J. Sel. Topics Signal Process*, vol. 5, pp. 805–816, May 2011.

[17] L. Xiao, S. Boyd, and S. Lall, "Distributed average consensus with time-varying metropolis weights," 2005, available online at http://web.stanford.edu/ boyd/papers/pdf/avg_metropolis.pdf.

[18] X. F. Zhu and M. Rabbat, "Approximating signals supported on graphs," in *IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Kyoto, Japan, Mar 2012, pp. 3921–3924.

[19] J.-D. Boissonnat and M. Yvinec, *Algorithms Geometry.* Cambridge: Cambridge University Press, 1998.

[20] J. Demmel, I. Dumitriu, and O. Holtz, "Fast linear algebra is stable," *Numerische Mathematik*, vol. 108, no. 1, pp. 59–91, Nov. 2007.

[21] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge University Press, 2004.

[22] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *J. Roy. Statistical Soc. Series C (Appl. Statist.)*, vol. 28, no. 1, pp. 100–108, 1979.

[23] F. A. Potra and S. J. Wright, "Interior-point methods," *J. Computational Appl. Mathematics.*, vol. 124, 2000.

[24] S. D. Gupta, M. Coates, and M. Rabbat, "Error propagation in gossip-based distributed particle filters," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, no. 3, pp. 148–163, Aug. 2015.

[25] A. Olshevsky and J. Tsitsiklis, "Convergence speed in distributed vonsensus and averaging," *SIAM J. Control Optimization*, vol. 48, pp. 33–55, Feb. 2009.