Graph-Based Compression for Distributed Particle Filters

Jun Ye Yu, Mark J. Coates, and Michael G. Rabbat

***Abstract*—A key challenge in designing distributed particle filters 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 very low communication overhead and provide a theoretical error bound for the presented filters.**

I. 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. Collaboration may involve dissemination of sensor measurements [1], [2], distributed computation of joint log-likelihoods [3]–[7], or propagation of posterior target distribution across sensors [8], and may require considerable communication overhead (e.g., ensuring the data reaches the furthest sensor, or reaching consensus on some network-wide statistics). Since the sensors are often battery-powered, a key objective in designing a distributed particle filter is to reduce the communication overhead without incurring major degradation of the tracking performance.

A. Distributed particle filters

A variety of architectures have been proposed for distributed particle filters and non-linear tracking over networks; see [9] for a detailed survey. In this paper, we focus on consensus-based algorithms. Each sensor maintains a local particle-representation of the target distribution and exchanges messages with neighboring sensors in a communication network. The objective is for all nodes to compute the 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 is to focus communication resources on a subset of particles. The set membership constrained filter [10] 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. A related approach, described in [11], proposes an auxiliary particle filter that uses selective gossip so only the particles with highest weights are communicated between sensors. Both methods improve over the naive approach, but they may still incur a high communication overhead if the likelihood distribution is not peaky and/or the number of particles is high.

A second approach is to approximate the posterior distribution as either a Gaussian distribution [3], [12] or as a mixture of Gaussians [13]. Sensors can first compute local approximations to their particle clouds, after accounting for their local observations. Then the sensors broadcast or reach consensus on 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 the joint log-likelihood of all particles. In the *likelihood consensus* method [4], the log-likelihood function is expressed through a linear transformation in terms of a known, pre-specified basis. The basis functions are assumed to be known to all sensors and they are sensor-independent. Each sensor computes 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. A trade-off between communication overhead and fidelity is controlled by truncating the basis expansion, with more terms, and hence coefficients, providing a more accurate approximation at the cost of higher communication overhead. This approach can achieve significant reduction in communication overhead when the approximation only requires a small number of coefficients (e.g., low-order polynomial functions). This approach has been further specialized to the particular setting where the sensors only observe bearings to the target, corrupted with additive Gaussian noise [7], [14].

B. Contribution

In this paper we present two novel approaches to distributed particle filtering. In contrast to prior approaches which expand the log-likelihood using a fixed basis (e.g., polynomial bases), we propose graph-based methods which adapt the basis functions to the current particle representation of the likelihood.

The first method we propose, called the *graph Laplacian particle filter*, fits a graph to the particles based on their position, and then expands the per-particle likelihoods (which

The authors are with the Department of Electrical and Computer Engineering, McGill University, 3480 University Street, Montréal, Québec, Canada. Email: jun.y.yu@mail.mcgill.ca, mark.coates@mcgill.ca, michael.rabbat@mcgill.ca

This work was supported in part by the Natural Sciences and Engineering Research Council of Canada grant DGDND-2017-00007

can be viewed as a signal over the graph) in terms of the Laplacian eigenbasis. Since most sensors induce smooth likelihood models (particles which are nearby in state space typically have similar likelihood values), truncating the representation to use just the first few Laplacian eigenvectors is expected to give a good approximation. After aggregating the coefficients over the network via gossip, each sensor can compute the joint log-likelihood by inverting the transformation in terms of the Laplacian eigenvectors. This approach gives a linear approximation since the expansion, to compute the coefficients, and the inversion, to recover the approximate joint log-likelihoods, are all linear operations.

To address the fact that computing an eigendecomposition may be computationally prohibitive or undesirable for low-power sensors, we introduce a second method, called the *cluster particle filter*. As the name suggests, sensors first partition the particles using a clustering algorithm, and they aggregate the log-likelihood of particles in the same cluster. Then the sensors gossip on these per-cluster coefficients. To recover per-particle log-likelihoods, each sensor interpolates by solving a convex optimization problem that takes the aggregate per-cluster coefficients as inputs. Since the solution to this interpolation problem is not linear in the inputs, the resulting scheme gives a non-linear approximation.

Preliminary results for the two proposed filters have been presented in [6] and [15]. In this paper, we present both filters in more detail, provide more comprehensive simulation results, and show that the proposed filters are able to yield robust tracking performance even with very low communication overhead. We also study a theoretical error propagation bound for the proposed filters to provide insights into their robustness. In addition, we propose a modification to the existing likelihood consensus method that improves its robustness when using few gossip iterations for distributed fusion.

C. Paper organization

The rest of the paper is organized as follows. Section II describes the tracking problem and provides an overview of distributed particle filtering. Section III reviews the likelihood consensus approach in more detail. Sections IV and V describe the proposed distributed particle filters. We derive a theoretical error bound in Section VI. Simulation results are provided in Section VIII. Finally, Section IX concludes the paper.

II. Problem formulation

A network of S sensors collaboratively track a single moving target over time. The target state evolves over time following a discrete-time Markov model:

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

where $X(k) \in \mathcal{X}$ is the target state at time step k, $\mathcal{X}$ denotes the state space (e.g., a Euclidean vector space), 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 k, each sensor s takes a measurement:

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

where h^s 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 $Z(k) = \big(z^1(k), ...z^S(k)\big)$ denote the measurements from all sensors at time k, and let $Z(1{:}k) = \big(Z(1), ..., Z(k)\big)$ denote all measurements from the first k time steps. The objective is to estimate the posterior target distribution $p(X(k)|Z(1{:}k))$ given all measurements up to time step k.

If the distribution $p(X(k{-}1)|Z(1{:}k{-}1))$ at time $k{-}1$ is available, then $p(X(k)|Z(1{:}k))$ can be obtained recursively in two steps via Bayes' rule. In the prediction step, we obtain a predicted density using the Chapman-Kolmogorov equation [16]:

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

Then, in the update step when new measurements $Z(k)$ become available, the predicted density is updated as [16]:

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

A. Centralized particle filter

Equations (3) and (4) form the basis of an optimal Bayesian solution, but in general they are computationally intractable to evaluate exactly. The particle filter [17] provides a feasible approximation by modeling the posterior distribution using a set of N weighted particles $\{X_i(k), w_i(k)\}_{i=1}^N$, where $X_i(k) \in \mathcal{X}$ are points in the state space, and $w_i(k)$ are non-negative weights satisfying $\sum_{i=1}^N w_i(k) = 1$. Using the weighted particles, we approximate the posterior as

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

where $\delta_X(x)$ is the Dirac delta function (i.e., 1 if $x = X$, and 0 otherwise).

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 $Z(k)$:

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

If the transition density $p(X_i(k)|X_i(k{-}1))$ is used as the proposal density, then the weight update simplifies to $w_i(k) \propto w_i(k{-}1)p(Z(k)|X_i(k))$.

Given the updated particle set $\{X_i(k), w_i(k)\}_{i=1}^N$, the *minimum-mean-squared-error* (MMSE) state estimate can be computed as

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

B. Distributed particle filters

Consider now a distributed implementation of the particle filter where each sensor maintains a local copy of weighted particles. Denote by $\{X_i^s(k), w_i^s(k)\}_{i=1}^N$ the set of weighted particles at sensor s. Assume that all sensors initialize their pseudo-random number generators using a common seed. The sensors are thus synchronized and generate the same sequence of random numbers at each time step. As a result, the sensors start with the same initial particles $\{X_i^s(1), w_i^s(1)\}_{i=1}^N$. Assume that the particles remain synchronized across sensors at time $k-1$ (i.e., $X_i^{s_1}(k-1) = X_i^{s_2}(k-1), w_i^{s_1}(k-1) = w_i^{s_2}(k-1), \forall i, s_1 \neq s_2$), each sensor then samples the same propagated particles from $q(X_i(k)|X_i(k-1), Z(1{:}K))$.

To maintain synchronization, all sensors need to agree on the posterior particle weights $w_i(k)$, $i = 1, \ldots, N$. Suppose that the transition density is used as the proposal density so that $w_i(k) \propto w_i(k-1)p(Z(k)|X_i(k))$. Since $w_i(k-1)$ is identical at all sensors, computing $w_i(k)$ amounts 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-likelihood factorizes:

$$\log(p(Z(k)|X_i(k))) = \sum_{s=1}^{S} \log(p(z^s(k)|X_i(k))). \quad (5)$$

Therefore, each sensor can compute $\log(p(z^s(k)|X_i(k))$ locally and obtain an approximation to the sum using gossip algorithms [18]–[20] or other consensus algorithms [21].

Gossip algorithms are distributed message passing algorithms for reaching a consensus on the sum or average of a collection of numbers, where initially each node in a network holds one of the numbers. Messages are passed between neighboring nodes (i.e., sensors) in the communication network, and nodes combine messages received by taking linear combinations. Under mild assumptions, the estimate at every node converges asymptotically to the average of the initial values [18], [20]. The rate of convergence depends on the weights used when combining messages received at each node, as well as the graph topology.

In any practical application, the gossip process will be truncated after a finite number of iterations, and hence there will be some disagreement between the values at different sensors and the average. In the context of distributed particle filters, this causes two issues. First, because the values are different at each sensor, the sensors will no longer be synchronized unless we perform additional processing. To remedy this, after running a fixed number of rounds of gossip for distributed averaging, we can run a max-consensus protocol [22], which is guaranteed to converge to the same value at all sensors after a finite number of communication rounds. If, after gossiping, all sensors have values which are close to the average (i.e., the errors are small), then the maximum of these values will also be close to the average.

The second issue caused by errors due to running gossip for a finite number of rounds is that the errors in log-likelihood calculations may accumulate over time, causing an overall error in the state estimates produced by the distributed particle filters running at each sensor. We studied this issue in our previous work [23], in which we proved that the error in the state estimate remains bounded over time as long as the relative error in the per-particle log-likelihoods remains bounded, and we showed how this could be achieved by controlling the number of gossip rounds as a function of the network topology in the simple setup where sensors gossip directly on the per-particle log-likelihoods. We will revisit this issue in Sec. VI in the context of the proposed distributed approximation schemes.

III. Likelihood consensus

A naive implementation of the distributed particle filter is to run one gossip algorithm per particle, at the cost of prohibitively high communication overhead. The *likelihood consensus particle filter* (LCpf) [4] seeks to reduce the communication overhead by approximating the log-likelihood function as a linear combination of m basis functions. We review likelihood consensus in more detail in this section.

Since the key step in distributed particle filters is the computation or approximation of the joint log-likelihood (5), and this is repeated at each time step, we omit the time step index k to simplify the notation for the remainder of the paper. In the experiments reported later in Sec. VIII, it should be understood that these operations are being performed in every tracking update step.

The LCpf is based on the approximation

$$\log p(z^s|X_i) \approx \sum_{j=1}^{m} \alpha_j^s \cdot \beta_j(X_i). \quad (6)$$

The basis functions $\beta_j(X_i)$ depend only on the particles X_i, and they are known to all sensors (i.e., the sensors are synchronized at the end of each step). The coefficients α_j^s, $j = 1, \ldots, m$, are computed locally at sensor s and depend on the local measurement z^s available only at sensor s.

Using this basis expansion, the joint log-likelihood (5) can be approximated as follows:

$$\begin{aligned}\gamma(X_i) &= \log p(Z|X_i) \\ &\approx \sum_{s=1}^{S}\sum_{j=1}^{m} \alpha_j^s \cdot \beta_j(X_i) \\ &= \sum_{j=1}^{m} \beta_j(X_i) \left(\sum_{s=1}^{S} \alpha_j^s\right).\end{aligned}$$

In other words, since the basis functions $\beta_j(X_i)$ are known to all sensors and can be computed locally, it suffices to compute the m aggregate coefficients $\sum_{s=1}^{S} \alpha_j^s$, $j = 1, \ldots, m$, to recover the joint log-likelihoods. When $m \ll N$, the communication overhead can be reduced significantly.

Let $\gamma^s(X_i) = \log(p(z^s|X_i))$ denote the log-likelihood of particle X_i at sensor s and let $\gamma_m^s(X_i) = \sum_{j=1}^{m} \beta_j(X_i)\alpha_j^s$ denote its approximate value. The local coefficients are chosen such that the sum of squared errors $\sum_{i=1}^{N} ||\gamma^s(X_i) - \gamma_m^s(X_i)||^2$ is minimized. More specifically, define matrix $\Psi \in \mathbb{R}^{N\times m}$ such that $\Psi(i,j) = \beta_j(X_i)$ and column vector $\gamma^s = [\gamma(X_1), ...\gamma(X_N)]^T$ where superscript T denotes the

transpose. Then the coefficients $\alpha^s = [\alpha_1^s, ...\alpha_m^s]^T$ can be computed as follows:

$$\alpha^s = (\Psi^T\Psi)^{-1}\Psi^T\gamma^s. \tag{7}$$

The approximate global log-likelihoods can be recovered from the aggregate coefficients as follows:

$$\gamma_m = \Psi\left(\sum_{s=1}^{S}\alpha^s\right). \tag{8}$$

We note that likelihood consensus is a non-adaptive, linear approximation since the expansion to compute the coefficients and the inversion to compute global log-likelihoods are all linear operations, and these coefficients are independent of the current particle cloud.

The original work on likelihood consensus [4] proposed to use low-order polynomials as the basis functions. Let $R_p \geq 0$ be the max polynomial degree. Without loss of generality, let $X = [x_1, ...x_d]$; i.e., the target state vector is d-dimensional. Then the approximate log-likelihood is computed as follows:

$$\gamma^s(X) \approx \sum_{r_1=0}^{R_p} ... \sum_{r_n=0}^{R_p} \alpha^s_{r_1,r_2,...,r_d} x_1^{r_1} x_2^{r_2} \cdots x_d^{r_d}. \tag{9}$$

The total number of basis functions is $m = (R_p+1)^d$ and each basis function has the form $\prod_{l=1}^{d} x_{i,l}^{r_l}$ with associated coefficient $\alpha^s_{r_1,...,r_d}$.

The *constraint sufficient statistics particle filter* (CSSpf) [5], which we include in the simulation comparison study in Sec. VIII, is a variant of the likelihood consensus filter and is specifically tailored for tracking a target in a two-dimensional state space using only bearing measurements corrupted with additive white Gaussian noise. This filter also applies a linear transformation to compress the log-likelihoods using a basis specifically tailored to this problem setting. Unlike the likelihood consensus filter with a polynomial basis, the number of basis functions is fixed at six regardless of the number of particles.

IV. The graph Laplacian particle filter

This section introduces a graph-based compression scheme which aims to reduce the communication overhead by exploiting similarity of log-likelihood values at nearby particles. The scheme can be viewed as a form of likelihood consensus [4] using a basis that adapts over time along with the particle cloud. In the experiments reported later in the paper, it should be understood that these operations are being performed in every tracking update step.

At a high level, the *graph Laplacian particle filter* (LApf) follows these steps:

1) Each sensor computes local particle log-likelihoods.
2a) Each sensor constructs a graph over the particles.
2b) Each sensor uses the eigenvectors of the resulting graph Laplacian to encode the log-likelihoods.
3) Gossip and max-consensus algorithms are used to compute the aggregate coefficients.
4) Each sensor recovers the joint particle log-likelihoods from the coefficients and the Eigenvectors.

Let us focus first on step 2. We consider each particle X_i as a vertex in a graph and construct a set of (possibly weighted) edges to connect these vertices. Let A denote the $N\times N$ weighted adjacency matrix where $A(i,j) = A(j,i) > 0$ if X_i and X_j are connected and 0 otherwise. 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 an eigendecomposition $L = \Psi\Lambda\Psi^T$ where Λ is a diagonal matrix of eigenvalues with corresponding eigenvectors given by the columns of Ψ. We assume that A is connected, by design; i.e., for any two particles X_s and X_t, there is a sequence $i_0, i_1, \ldots, i_l$ with $i_0 = s$ and $i_l = t$, such that $A(i_{j-1}, i_j) > 0$ for all $j = 1, \ldots, l$. Consequently, the smallest eigenvalue of L is $\lambda_1 = 0$ and all other eigenvalues are strictly positive [24]. We assume that the eigenvalues are sorted in ascending order, so $\lambda_1 = 0 < \lambda_2 \leq \cdots \leq \lambda_N$, and the ith column of Ψ is the eigenvector with corresponding eigenvalue λ_i.

The Laplacian eigenvectors can be used as a Fourier-like basis [25] for signals supported on the graph. Using all N eigenvectors for the Fourier transform would be counter-productive since we achieve no reduction in communication overhead. Instead, we achieve compression by projecting onto only $m \ll N$ 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.

The log-likelihood vector $\gamma^s = [\gamma^s(X_1), ..., \gamma^s(X_N)]^T$ can be encoded as follows:

$$\alpha^s = \Psi_m^T\gamma^s \tag{10}$$

where Ψ_m is a matrix consisting of the m column eigenvectors corresponding to m smallest eigenvalues.

In the multi-sensor setting, m gossip algorithms are run in parallel to compute the coefficients $\alpha_j = \sum_s \alpha_j^s$, $j = 1, \ldots, m$.[1] The number of coefficients m thus represents a trade-off between communication overhead and compression error (i.e., discrepancy between γ^s and γ_m^s, the log-likelihoods reconstructed from m coefficients). In the limit case of $m = N$, there is no compression error but no reduction in communication overhead either. This parameter can be fixed by users beforehand or set dynamically over time by individual sensors. In the latter case, each sensor selects a suitable value of m to ensure that the discrepancy $\sum_{i=1}^{N} |\gamma^s(X_i) - \gamma_m^s(X_i)|$ (or other suitable distance metric) is sufficiently low. Then a max-consensus algorithm can be run to determine the maximum m among all sensors.

Finally, let $\widehat{\alpha} \approx \sum_s \alpha^s$ denote the vector of coefficients obtained after gossiping in Step 3. In Step 4, the vector of per-particle log-likelihoods can be recovered via the inverse transform,

$$\gamma_m = \Psi_m\widehat{\alpha} \approx \Psi_m\Psi_m^T\sum_s \gamma^s. \tag{11}$$

[1] Alternatively, one can see this as gossiping on a vector of dimension m.

We note that the eigenvalue decomposition imposes a heavy computational burden on the sensors. In general, it has computational complexity $O(N^3)$ [26], although it may be possible to use methods (such as Lanczos iterations [27] or preconditioned conjugate gradient [28]) which exploit the sparsity of L to compute a small number eigenvectors more efficiently.

V. 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) described in this section exploits smoothness via the graph Laplacian without computing the eigenvalue decomposition of the Laplacian matrix.

At a high level, the Clusterpf follows steps similar to the graph Laplacian particle filter:

1) Each sensor computes local particle log-likelihoods;
2a) Each sensor groups the particles into K clusters;
2b) Each sensor computes the cluster log-likelihood;
3) Gossip and max-consensus algorithms are used to compute the joint cluster log-likelihoods;
4) Each sensor interpolates the joint particle log-likelihoods from the joint cluster log-likelihoods.

Consider Step 2. We apply K-means clustering to group the particles into K clusters based on their proximity to each other. Other methods such as spectral clustering [29] may also be considered. Let C denote the $K\times N$ cluster assignment matrix where $C(i,j) = 1$ if particle j belongs to cluster i. By assumption, the particle values X_i^s at all sensors are synchronized at the end of each iteration, and sensors have synchronized seeds for their random number generators. Hence the resulting cluster matrix C may be taken to be identical at all sensors too.

Rather than projecting γ onto Laplacian eigenvectors, for compression we aggregate the log-likelihood values of all particles in the same cluster and then gossip on one coefficient per cluster. The log-likelihood of the i^{th} cluster, γ_c^i, is equal to the sum of the log-likelihoods of its constituent particles. We can thus relate the cluster log-likelihoods to the particle log-likelihoods via

$$\gamma_c = C\gamma, \tag{12}$$

where $\gamma_c \in \mathbb{R}^K$ is the compressed coefficient vector. Rather than gossiping on the $N\times 1$ vector γ^s, each sensor only needs to gossip on the $K\times 1$ vector $\gamma_c^s = C\gamma^s$. When $K \ll N$, significant reduction in communication overhead can be achieved. Let $\widehat{\gamma}_c \approx \sum_s \gamma_c^s$ denote the result of running gossip and max-consensus on the per-sensor coefficient vectors γ_c^s in Step 3.

In Step 4, given $\widehat{\gamma}_c$, 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 since the resulting log-likelihoods exhibit sharp changes at cluster boundaries. Instead, we exploit the graph Laplacian to ensure that the log-likelihood values remain smooth over the state space. We again construct a graph over the particles, compute the Laplacian matrix L, and then solve the following convex quadratic programming problem:

$$\begin{array}{ll} \underset{\gamma}{\text{minimize}} & \frac{1}{2}\gamma^T L\gamma \\ \text{subject to} & C\gamma = \widehat{\gamma}_c. \end{array} \tag{13}$$

In other words, we find particle log-likelihood values that are smooth with respect to particle proximity, where we use the Laplacian quadratic form as our measure of smoothness, while ensuring that the aggregate values are consistent with the cluster values computed over the network. Since the matrix L is positive semi-definite, the problem is convex and can be solved using well-known methods [30].

A. Solution Uniqueness

Although the quadratic program defined in (13) is convex, one may still wonder if it has a unique solution since the graph Laplacian L is positive semi-definite. Next we show that it indeed has a unique solution.

The Lagrangian function for the constrained problem (13) is

$$\mathcal{L}(\gamma,\nu) = (1/2)\gamma^T L\gamma + \nu^T(C\gamma - \widehat{\gamma}_c),$$

where $\nu \in \mathbb{R}^K$ is a vector of Lagrange multipliers. The KKT conditions for this problem state that if $\gamma^\star$ is a solution to (13) then $\gamma^\star$ is feasible, so $C\gamma^\star = \widehat{\gamma}_c$, and there exists a Lagrange multiplier vector $\nu^\star$ such that [31]

$$\nabla_\gamma \mathcal{L}(\gamma^\star,\nu^\star) = L\gamma^\star + C^T\nu^\star = 0.$$

These conditions can be expressed simultaneously as the system of linear equations

$$\begin{bmatrix} L & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \gamma^\star \\ \nu^\star \end{bmatrix} = \begin{bmatrix} 0 \\ \widehat{\gamma}_c \end{bmatrix}. \tag{14}$$

To show that (13) has a unique solution, we will show that the matrix on the left of (14) is non-singular.

First, recall that the constructed graph over the particles is connected, by design. Let $\mathbf{1}_N$ denote an N-dimensional vector with all entries equal to 1. For a connected graph, it is well known that the null space of its Laplacian matrix L is spanned by the constant vector $\mathbf{1}_N$; i.e., $L\gamma = 0$ if and only if $\gamma = \alpha\mathbf{1}_N$ for some scalar a. Thus, 0 is an eigenvalue of L with corresponding (normalized) eigenvector $(1/\sqrt{N})\mathbf{1}_N$. Moreover, all other eigenvalues of L are strictly positive.

Recall, also, that each row of C is an indicator vector for one cluster, and the clusters partition the particles by design. Hence, the rows of C are linearly independent, and $C^T\mathbf{1}_K = \mathbf{1}_N$. Let $Z \in \mathbb{R}^{N\times(N-K)}$ be a matrix whose columns span the null space of C; i.e., $CZ = 0$, and if $C\gamma = 0$ then $\gamma = Z\beta$ for some vector $\beta \in \mathbb{R}^{N-K}$. Since $\mathbf{1}_N^T = \mathbf{1}_K^T C$, it follows that $\mathbf{1}_N^T Z = 0$; i.e., the constant vectors are not in the null space of C. Therefore $Z^T LZ$ is positive definite.

Now, let γ and ν be vectors for which

$$\begin{bmatrix} L & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \gamma \\ \nu \end{bmatrix} = 0. \tag{15}$$

Then $C\gamma = 0$. Consequently, from

$$\begin{bmatrix} \gamma \\ \nu \end{bmatrix}^T \begin{bmatrix} L & C^T \\ C & 0 \end{bmatrix} \begin{bmatrix} \gamma \\ \nu \end{bmatrix} = 0$$

it follows that $\gamma^T L\gamma = 0$. Since $C\gamma = 0$, there exists $\beta \in \mathbb{R}^{N-K}$ such that $\gamma = Z\beta$, and therefore $\beta^T Z^T LZ\beta = 0$. However, since $Z^T LZ$ is positive definite, it must be that $\beta = 0$, and so $\gamma = 0$. From (15), then, it must be that $C^T\nu = 0$, and this can only be true if $\nu = 0$ since C has full row rank. Therefore, since (15) holds only if $\gamma = 0$ and $\nu = 0$, we have shown that the matrix on the left-hand side of (14) is non-singular, and hence (13) has a unique solution.

B. Computational considerations

The KKT conditions (14) suggest that one way to find a solution to (13) is by directly solving this $(N+K)$-by-$(N+K)$ linear system. Direct matrix inversion has a computational complexity of $\mathcal{O}\big((N+K)^K\big)$ in general. However, since the coefficient matrix in this system is very sparse (recall that L is the Laplacian matrix of a graph, and C has exactly N non-zeros), the system can be solved (approximately, or exactly) more efficiently using an iterative indirect method such as preconditioned conjugate gradient [31].

C. Connection to graph Laplacian compression

We now make a connection to the graph Laplacian particle filter from Sec. IV to further motivate the use of the Cluster particle filter method. Recall that in the graph Laplacian particle filter, we compute coefficients $\alpha^s = \Psi_m^T \gamma^s$ at each sensor, where Ψ_m is the matrix formed from the first m Laplacian eigenvectors. After summing over sensors, we have $\widehat{\alpha} \approx \sum_s \alpha^s = \Psi_m^T \gamma^s$.

Now, consider the quadratic program,

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\gamma^T L\gamma \\ \text{subject to} & \Psi_m^T\gamma = \widehat{\alpha}. \end{array}$$

It is straightforward to show that the minimizer is given by $\hat{\gamma} = \Psi_m\alpha$, which is the same linear recovery method used in graph Laplacian particle filtering. From this perspective, we can view the cluster particle filter as substituting the clustering matrix C in place of the first m Laplacian eigenvectors. While this substitution is motivated from a computational perspective (clustering the particles can be significantly faster than computing the Laplacian eigenvectors when N is large), we may expect the two approaches to provide similar performance when C and Ψ_m provide similar information, and indeed well-known connections between Laplacian eigenvectors and spectral clustering algorithms [29] suggest that this should be the case.

VI. Error analysis

Using a weighted particle cloud to approximate the posterior introduces error in Bayesian filtering relative to the optimal, but intractable, update equations (3) and (4). Using gossip algorithms and other approximations, such as projecting onto the subspace spanned by a few Laplacian eigenvectors, or cluster-based compression, introduces additional errors. It is natural to ask whether these errors may accumulate over time.

For gossip-based distributed particle filters, this question is studied in [23] and a general framework is developed, building on ideas set forth in [32], under which one can prove that the error between the (intractable) optimal centralized Bayesian filter and a gossip-based distributed implementation remains uniformly bounded over time.

In the theory of [23], the error specifically due to decentralization and gossip can be controlled by ensuring that the relative error in the log-likelihood of each particle remains strictly less than 1 at every time step. More specifically, 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. Then a key condition in [23] is to ensure that

$$\frac{||\hat{\gamma}(X_i) - \gamma(X_i)||}{||\gamma(X_i)||} < 1$$

at every time step. The other conditions are:

- The Markov chain associated with target state transition undergoes sufficient mixing within a finite number of time steps;
- The state estimate function $F(X_i)$ can be suitably scaled and bounded such that $\sup_{X_i} |F(X_i)| < 1$; and
- The particle likelihood is bounded such that $\exp(\gamma(X_i)) \leq 1$ for all $i = 1, \ldots, N$.

Note that fulfillment of the first two conditions is unrelated to the choice of the particle filter. The third condition can be easily satisfied with suitable normalization of log-likelihoods. Thus our discussion focuses only on bounding the relative error $||\hat{\gamma}(X_i) - \gamma(X_i)||/||\gamma(X_i)||$.

A. General bound for distributed particle filters

We will first derive an error bound for general gossip-based distributed particle filters. Then we will specialize this bound to the graph Laplacian particle filter.

Recall that the distributed filters presented in the previous sections all involve the 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 gossip iterations.

Consider the encoding error. Let $\gamma_m(X_i)$ denote the approximate log-likelihood of particle X_i recovered using m coefficients; i.e., after exactly aggregating the per-sensor coefficients from Step 2, or equivalently, after running an infinite number of gossip iterations in Step 3. For a given approximation scheme (e.g., using Laplacian eigenvectors, clusters, or another non-adaptive approach), let $\delta_m > 0$ denote the smallest value for which

$$|\gamma_m(X_i) - \gamma(X_i)| \leq \delta_m |\gamma(X_i)|. \tag{16}$$

is guaranteed to hold for all $i = 1, \ldots, N$. The value of δ_m depends on the specific approximation scheme and the number of coefficients m used, and we expect δ_m to decrease as m increases. Note that Eq. (16) can be rewritten to yield

$$(1 - \delta_m)|\gamma(X_i)| \leq |\gamma_m(X_i)| \leq (1 + \delta_m)|\gamma(X_i)|. \quad (17)$$

Next, consider the gossiping error. Let $\hat{\alpha}$ denote the coefficients obtained after gossiping and max-consensus iterations. This gives us the approximate log-likelihoods $\hat{\gamma}$. Given $\delta_{gossip}>0$, we can run a sufficient number of gossip iterations to ensure that the following bound holds for all coefficients [23]:

$$\frac{|\gamma_m(X_i) - \hat{\gamma}(X_i)|}{|\gamma_m(X_i)|} \leq \delta_{gossip}. \quad (18)$$

The number of gossip iterations required to ensure that the relative error is at most δ_{gossip} depends on the sensor network topology and the initial values α^s at each sensor [23].

Putting everything together, we obtain

$$\begin{aligned}
\delta &= \frac{|\hat{\gamma}(X_i) - \gamma(X_i)|}{|\gamma(X_i)|} \\
&= \frac{|\hat{\gamma}(X_i) - \gamma_m(X_i) + \gamma_m(X_i) - \gamma(X_i)|}{|\gamma(X_i)|} \\
&\leq \frac{|\hat{\gamma}(X_i) - \gamma_m(X_i)|}{|\gamma(X_i)|} + \frac{|\gamma_m(X_i) - \gamma(X_i)|}{|\gamma(X_i)|} \\
&\leq (1 + \delta_m)\frac{|\hat{\gamma}(X_i) - \gamma_m(X_i)|}{|\gamma_m(X_i)|} + \frac{|\gamma_m(X_i) - \gamma(X_i)|}{|\gamma(X_i)|} \\
&\leq (1 + \delta_m)\delta_{gossip} + \delta_m \quad (19)
\end{aligned}$$

From (19), we see that the upper bound of δ exceeds 1 if either $\delta_m \geq 1$ or $\delta_{\text{gossip}} \geq 1$. We thus need to choose a suitable number of coefficients and sufficient gossip iterations to control both errors. The optimal combination of the parameters depends on the specific tracking problem set-up. We explore the tradeoff between these parameters experimentally in Sec. VIII by varying the number of Laplacian expansion coefficients or clusters (which affects δ_m) and the number of gossip iterations (which affects δ_{gossip}).

B. Controlling gossiping error of filters using linear transform coding

The general bound (19) applies to both the graph Laplacian particle filter and the cluster particle filter. The precise values of δ_m and δ_{gossip} may differ for the two filters since they use different encoding and decoding schemes.

In this section we focus specifically on bounding the gossiping error for methods that use linear transform coding; i.e., methods where the encoding used in Step 2 is a linear transformation of the per-particle log-likelihoods at each sensor, and the reconstruction in Step 4 is a linear transformation of the aggregate coefficients obtained by gossiping. The graph Laplacian particle filter is one such method; likelihood consensus with polynomial bases is another example. The bound derived below does not apply for the cluster particle filter because it uses a nonlinear recovery scheme.

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:

$$\frac{|\hat{\alpha}_j(l) - \alpha_j|}{|\alpha_j|} \leq \tau, \quad 1 \leq j \leq m. \quad (20)$$

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. (20) holds if the minimum number of gossip iterations satisfies the following condition [33]

$$l \geq \frac{1.5\log(S) + \log(\frac{S-1}{\tau})}{\log(1/\rho_W)}. \quad (21)$$

Thus, depending on the properties of the sensor communication topology (as captured by ρ_W), and the required accuracy τ, we can calculate the worst-case number of gossip iterations to run. Since this bound ensures that (20) holds at all agents, and for all coefficients, taking the maximum over sensors (via max-consensus) to ensure that the sensors remain synchronized does not make this error any larger. Max consensus is guaranteed to converge in a finite number of iterations (typically on the order of the network diameter).

Now, suppose that the encoding step is based on a linear transformation, so $\alpha^s = (\Psi^T\Psi)^{-1}\Psi^T\gamma^s$, where Ψ is a $N \times m$ matrix, and let $\hat{\alpha} \approx \sum_s \alpha^s$ denote the result of gossiping. The linear recovery scheme uses $\hat{\gamma} = \Psi\hat{\alpha}$. Similarly, let $\gamma_m = \Psi\alpha$, where $\alpha = \sum_s \alpha^s = (\Psi^T\Psi)^{-1}\Psi^T\gamma$, denote the result of encoding and decoding the aggregate log-likelihoods directly (with exact aggregation). Note that, for the graph Laplacian particle filter, the columns of Ψ are (normalized) eigenvectors, so $\Psi^T\Psi = I$. We write the encoding operation in terms of the pseudo-inverse $(\Psi^T\Psi)^{-1}\Psi^T$ instead of Ψ^T directly to also be consistent with likelihood consensus.

To derive an upper bound for the gossiping error, observe that

$$\begin{aligned}
\frac{|\hat{\gamma}(X_i) - \gamma_m(X_i)|}{|\gamma_m(X_i)|} &\leq \frac{||\hat{\gamma} - \gamma_m||_\infty}{|\gamma_m(X_i)|} \\
&= \frac{||\Psi(\hat{\alpha} - \alpha)||_\infty}{|\gamma_m(X_i)|} \\
&\leq \frac{||\Psi||_\infty||\hat{\alpha} - \alpha||_\infty}{(1 - \delta_m)|\gamma(X_i)|} \\
&\leq \frac{\tau||\Psi||_\infty||\alpha||_\infty}{(1 - \delta_m)\min_{X_i}|\gamma(X_i)|} \quad (22)
\end{aligned}$$

where the third line follows from the definition of the matrix ∞-norm and (16), and the last line follows from (20).

C. Likelihood consensus / graph Laplacian comparison

In the ideal case of $\delta_m = 0$ (i.e., perfect reconstruction of log-likelihoods from m coefficients), the denominator of the bound (22) is the same for all algorithms. Consider the numerator. Higher error τ for aggregate coefficient increases the upper bound as we would expect. Different filters have different transformation matrices Ψ and corresponding coefficients α which also impact the error bound. For the graph

Laplacian (LA) and likelihood consensus (LC) particle filters to achieve the same upper bound in Eq. (22), we should have

$$\frac{\tau^{LC}}{\tau^{LA}} = \frac{||\Psi^{LA}||_\infty ||\alpha^{LA}||_\infty}{||\Psi^{LC}||_\infty ||\alpha^{LC}||_\infty}. \quad (23)$$

For the graph Laplacian particle filter, the matrix Ψ^{LA} contains the m orthonormal eigenvectors corresponding to the m smallest eigenvalues of the graph Laplacian. Therefore, we have $|\Psi^{LA}(i,j)| \leq 1$ and $||\Psi^{LA}||_\infty \leq m$. In contrast, for likelihood consensus with polynomial basis functions (see (9)), $\Psi^{LC}(i,j) = \beta_j(X_i) = x_{i,1}^{r_1} x_{i,2}^{r_2} ... x_{i,d}^{r_d}$ where $x_{i,l}$ is the l^{th} component of X_i and the index j is mapped to a vector $\{r_1, ..., r_d\} \subseteq \{0, ..., R_p\}^d$. Depending on the maximum polynomial degree R_p and the individual particle X_i, we have $1 \leq \Psi^{LC}(i,j) \leq x_{i,1}^{R_p} x_{i,2}^{R_p} ... x_{i,d}^{R_p}$ and consequently $||\Psi^{LC}||_\infty \gg ||\Psi^{LA}||_\infty$. Deriving a theoretical bound for $||\alpha||_\infty$ or $\|\Psi\|_\infty$ would require making much strong assumptions about the system model, and is beyond the scope of this work. We will compare $||\alpha^{LC}||_\infty$ and $||\alpha^{LA}||_\infty$ empirically in Sec. VIII and see that $||\Psi^{LC}||_\infty ||\alpha^{LC}||_\infty \gg ||\Psi^{LA}||_\infty ||\alpha^{LA}||_\infty$. This, combined with Eq. (23), suggests that likelihood consensus should require more gossip iterations to reduce τ accordingly in order to achieve the same gossiping error bound as the graph Laplacian particle filter.

VII. Likelihood Consensus with Gram-Schmidt

We validate the error bounds derived in the previous section via experiments reported in Sec. VIII. There we show that likelihood consenus does indeed require more gossip iterations to achieve similar tracking performance as the graph Laplacian particle filter.

In the previous section, we showed that the overall error of the distributed filter is upper-bounded by the expression in (19), which involves both the encoding error and the gossiping error. Furthermore, the upper bound of the gossiping error depends on $||\Psi||_\infty ||\alpha||_\infty$. In the experiments reported below, we observed that this quantity can be much larger for likelihood consensus than for the graph Laplacian approximation. Subsequent investigation suggested that this may be related to the matrix Ψ used for polynomial bases in likelihood consensus. In general, Ψ may have a high norm.

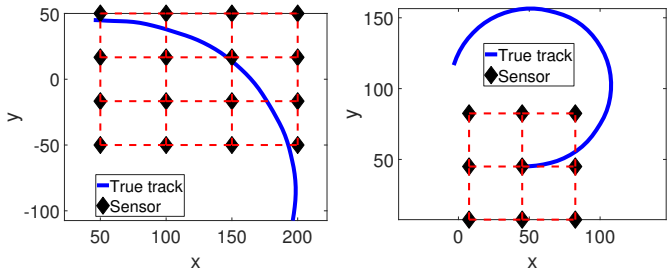
To reduce this error bound, we propose a modification to the likelihood consensus algorithm whereby each sensor runs the Gram-Schmidt procedure on Ψ so that the resulting columns used are all mutually orthogonal, have unit norm, and thus play a role similar to the Laplacian eigenvectors. Experimentally, in the next section, we observe that this modification reduces the gossiping error. Consequently running likelihood consensus with an orthonormal matrix obtained via Gram-Schmidt is expected to have better performance than standard likelihood consensus at low communication overhead, at the cost of additional computational overhead (to run the Gram-Schmidt procedure).

VIII. Numerical Study

A. Experimental setup

In this section, we evaluate the performance of the graph Laplacian particle filter (LApf) and the cluster particle filter (Clusterpf). For comparison, we also evaluate the likelihood consensus particle filter (LCpf) [4], constraint sufficient statistics particle filter (CSSpf) [5], Gaussian approximation particle filter (GApf) [3] and include a centralized bootstrap particle filter (BSpf) as a baseline. We also run the likelihood consensus particle filter with Gram-Schmidt orthogonalization (LCpf-GS) described in Sec. VII.

We consider two different simulated tracks with different measurement models (see Fig. 1). In each scenario a network of sensors collaboratively track a moving target over 50 time steps.



(a) Track 1 (b) Track 2

Fig. 1. 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 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 coordinate velocities. The state evolves according to

$$X(k+1) = F(X(k)) + \xi(k) \quad (24)$$

where $F(X(k))$ is the dynamic model and $\xi(k)$ is 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 the constant velocity model, we have

$$F(X(k)) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (25)$$

For the coordinated turn model, we have

$$F(X(k)) = \begin{bmatrix} 1 & 0 & \frac{\sin(\Omega)}{\Omega(k)} & -\frac{1-\cos(\Omega(k))}{\Omega(k)} \\ 0 & 1 & \frac{1-\cos(\Omega(k))}{\Omega(k)} & \frac{\sin(\Omega(k))}{\Omega(k)} \\ 0 & 0 & \cos(\Omega(k)) & -\sin(\Omega(k)) \\ 0 & 0 & \sin(\Omega(k)) & \cos(\Omega(k)) \end{bmatrix}. \quad (26)$$

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

$$\Omega(k) = \frac{a}{\sqrt{\dot{x}^2(k) + \dot{y}^2(k)}} \quad (27)$$

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

We assume that the sensor positions are fixed, and that sensor s knows its own position, which is denoted by $[x^s, y^s]$.

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

$$H_s(X(k)) = \arctan 2\left(\frac{x_t(k) - x_s}{y_t(k) - y_s}\right) + \eta(k) \quad (28)$$

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

$$H_s(X(k)) = \sqrt{(x_t(k) - x_s)^2 + (y_t(k) - y_s)^2} + \eta(k). \quad (29)$$

The process noise $\xi(k)$ has covariance matrix Q [34] equal to:

$$Q = \sigma_a^2 \begin{bmatrix} \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 1 & 0 \\ 0 & \frac{1}{2} & 0 & 1 \end{bmatrix} \quad (30)$$

where $\sigma_a = 10^{-4}$, and the measurement noise $\eta(k)$ has variance $R = \sigma^2$ where, depending on the scenario, $\sigma_\theta = 0.0873$ rad $= 5$ degree for bearing measurements, and $\sigma_r = 5$ km for range measurements.

All particle filters use the same number, N, of particles, which we will vary in the simulations. At time step 1, we generate an initial vector $X_{init} \sim \mathcal{N}(X(1), R_{\text{initial}})$ with $R_{\text{initial}} = \text{diag}([0.5^2, 0.5^2, 0.05^2, 0.05^2])$. We then generate the initial particles $X_i(1) \sim \mathcal{N}(X_{init}, R_{\text{particles}})$ with $R_{\text{particles}} = \text{diag}([5^2, 5^2, 0.5^2, 0.5^2])$. We adopt this two-step approach to give all filters an inaccurate initial state estimate.

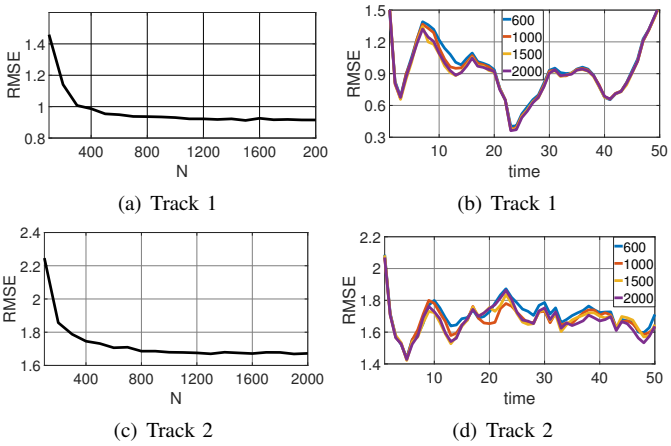
For LCpf, we use the polynomial basis functions [4] with max polynomial degree R_p, but we only include each particle's position in the basis function since the likelihood function (bearing and range) does not depend on the particles' velocity. In other words, we use all permutations of $x_t^i y_t^j$ with $0 \leq i, j \leq R_p$ (i.e., For $R_p = 2$, the basis functions are $\beta_1(X) = x_t^0 y_t^0 = 1$, ..., $\beta_9(X) = x_t^2 y_t^2$.). For LApf, we retain $m \leq N$ eigenvectors as the basis of Laplacian transformation. For Clusterpf, all particles are grouped into K clusters.

The random number generators are synchronized to ensure that the particles remain the same across sensors. Distributed summation is performed using the gossip algorithm [18]. 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:

$$g_{i,\text{new}} = w_{ii} g_{i,\text{old}} + \sum_{j \in N_i} w_{ij} g_{j,\text{old}} \quad (31)$$

$$w_{ij} = \begin{cases} \frac{1}{1+\max(|N_i|,|N_j|)} & j \in N_i \\ 1 - \sum_{j \in N_i} w_{ij} & i = j \\ 0 & j \notin N_i \end{cases} \quad (32)$$

where N_i denotes the set of neighboring sensors of sensor i and Metropolis weight [21] is used for the update. Since only a finite number of gossip iterations are executed, it is not guaranteed that all sensors will obtain the same values. Therefore, a max-consensus algorithm is executed to ensure all sensors obtain the same values. Define an update matrix W where $W(i,j) = w_{ij}$. Given initial values $g_{\text{initial}} = [g_1, ..., g_S]^T$, the final value is equivalent to $g_{\text{final}} = \max(W^{NGossip} g_{\text{initial}})$.



(a) Track 1 (b) Track 1

(c) Track 2 (d) Track 2

Fig. 2. RMSE of bootstrap particle filter with respect to N. All results are averaged over 200 random Monte-Carlo trials.

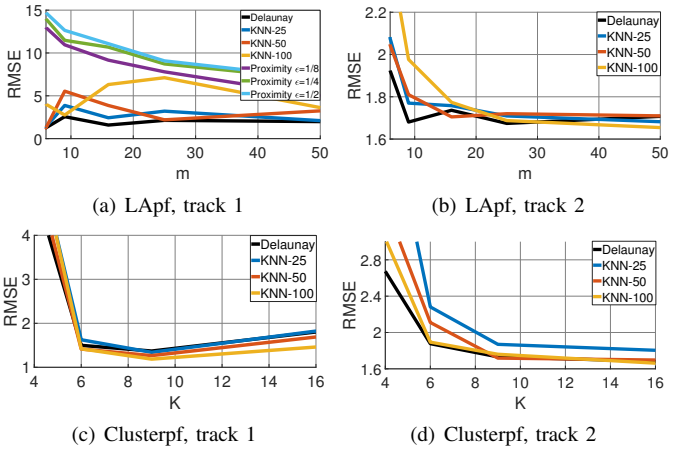
In the remainder of the section, we run a number of Monte Carlo simulations to evaluate the performance of all filters. The track and the sensor positions remain the same in each trial; but the measurements differ. We evaluate the algorithms' performances using the *root mean squared error* (RMSE) of position estimate. Note that CSSpf is not run for track 2 since it is designed for bearings-only tracking.

B. Centralized baseline, varying N

Fig. 2 shows the average RMSE of BSpf with respect to N to provide a baseline for the optimal tracking performance. The average RMSE decreases with increasing N as expected. The RMSE of track 2 is higher than that of track 1. This is to be expected since, in track 2, there are fewer sensors and the target is moving further away from the sensors over time. For track 1, the RMSE fluctuates over the entire duration of the track. For track 2, the RMSE fluctuates around 1.7 after time step 10 (when the target goes outside the sensor grid). For both tracks, we do not see a significant reduction in RMSE for $N \geq 1000$. For the rest of the paper, unless otherwise stated, we choose $N = 1000$ for all our simulations.

C. Graph construction

Both LApf and Clusterpf require constructing a graph over the particles. We compare three different approaches to constructing the particle similarity graph: *K-nearest-neighbor* (KNN) graphs, *Delaunay triangulation* (DT) graph [35] and proximity graph. In the first graph, each particle is connected to its K nearest neighbors. The second graph induces a triangulation on the particles such that no particle is inside the circumcircle of any triangle in the graph. In a proximity graph, each particle is connected to all other particles that are at most a distance of r away. We run LApf and Clusterpf on both tracks using all three graph construction methods. For the proximity graph, we set the connectivity radius as follows. Let d_{max} denote the maximum Euclidean distance between any two particles. The connectivity radius is set to $r = \epsilon d_{max}$ with



(a) LApf, track 1 (b) LApf, track 2

(c) Clusterpf, track 1 (d) Clusterpf, track 2

Fig. 3. RMSE of LApf and Clusterpf with respect to m (LApf) and K (Clusterpf) using different graph construction methods. For DT graph, no additional parameter is required. For KNN graph, we vary the number of neighbors each particle is connected to. For the proximity graph, we vary ϵ which determines the connectivity radius ϵd_{max} where d_{max} is the maximum Eucliean distance between any two particles. All results are averaged over 200 random trials.

$0 < \epsilon \leq 1$. We note that all three graph algorithms consider only the particles' positions since the likelihood functions (bearing and range) do not depend on the particles' velocities.

In this set of simulations, the aggregate coefficients are computed without gossiping so all sensors obtain the true aggregate coefficients; thus, for a given number of eigenvectors/clusters, any difference in performance is due solely to the method used to construct the particle graph. To generate the Laplacian matrix, we define the weighted adjacency matrix A. If particles X_i and X_j are connected, then $A(i,j) = A(j,i) = 1/\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$. Otherwise, $A(i,j) = A(j,i) = 0$. We also experimented with using adjacency weights equal to the inverse distance squared, $A(i,j) = A(j,i) = 1/\left((x_i - x_j)^2 + (y_i - y_j)^2\right)$, exponential weights $A(i,j) = A(j,i) = \exp\left(-((x_i - x_j)^2 + (y_i - y_j)^2)\right)$, and unweighted graphs (i.e., $A(i,j)$ either 0 or 1), but the inverse distance weights consistently led to the best performance.

For each graph, we compute the 2-norm total graph variation [25], defined as $||V||_G = (\gamma^T L_G \gamma)^{1/2}$, where G denotes the particle graph, L_G is the corresponding Laplacian matrix, and γ is the column vector containing the particle log-likelihoods. This metric characterizes the smoothness of the signal defined on the graph with a lower value indicating a smoother signal (and better compressibility). We find that the signal is most smooth on the Delaunay triangulation graph which can translate to lower RMSE when the signal is compressed using a fixed number of coefficients. We omit the relevant figure due to space constraint.

We study the average RMSE with respect to m for LApf and K for clusterpf. Fig. 3 shows the results. For both LApf and Cluserpf, the proximity graph consistently leads to much worse performance (see Fig. 3 (a) for example). We omit the proximity graph curves in the remaining figures (since the same trends persist) and focus our discussion on Delaunay triangulation and KNN graphs.

The RMSE decreases with increasing m or K for both graph methods as expected. The Delaunay graph consistently yields very competitive RMSE in both tracks at low communication overhead. Furthermore, the Delaunay triangulation graph does not require tuning any parameter (i.e., k or ϵ). We also studied the normalized particle weights discrepancy $||w_{\text{true}} - w_{\text{approx}}||_2$ and observed the same trends as for RMSE, so we omit the figures. We thus use the Delaunay triangulation graph with inverse distance weights for LApf and Clusterpf in the remaining simulations. Note that, although the Delaunay triangulation is well-defined for particles in any dimension, it is only efficient to compute in 2-D and 3-D. For higher-dimensional settings the KNN graph construction may be preferable.

D. Number of approximation terms

The parameter m of LApf offers a trade-off between communication overhead and approximation error of log-likelihoods. We expect that using a larger m should give a better approximation, but at the cost of higher communication overhead. The parameter K of Clusterpf plays a similar role. We run LApf and Clusterpf on both test tracks using various combinations of m/K and NGossip. For LCpf, the number of coefficients is $(R_p+1)^2$ where R_p is the maximal polynomial degree for the basis functions. We thus choose m/k values so that LApf/Clusterpf have the same coefficients as LCpf at $R_p = 1, 2, ..., 5$ (i.e., $m/K = 4, 9, 16, 25, 36$). Fig. 4 shows the average RMSE with respect to total number of scalars transmitted per sensor per time step.

Consider LApf. For both tracks, increasing NGossip improves tracking performance for all values of m. For track 1, setting $m = 6$ yields low RMSE at low communication overhead and yields the lowest RMSE at higher overhead by a significant margin. For track 2, setting $m = 9$ seems to offer a good balance between communication overhead and RMSE.

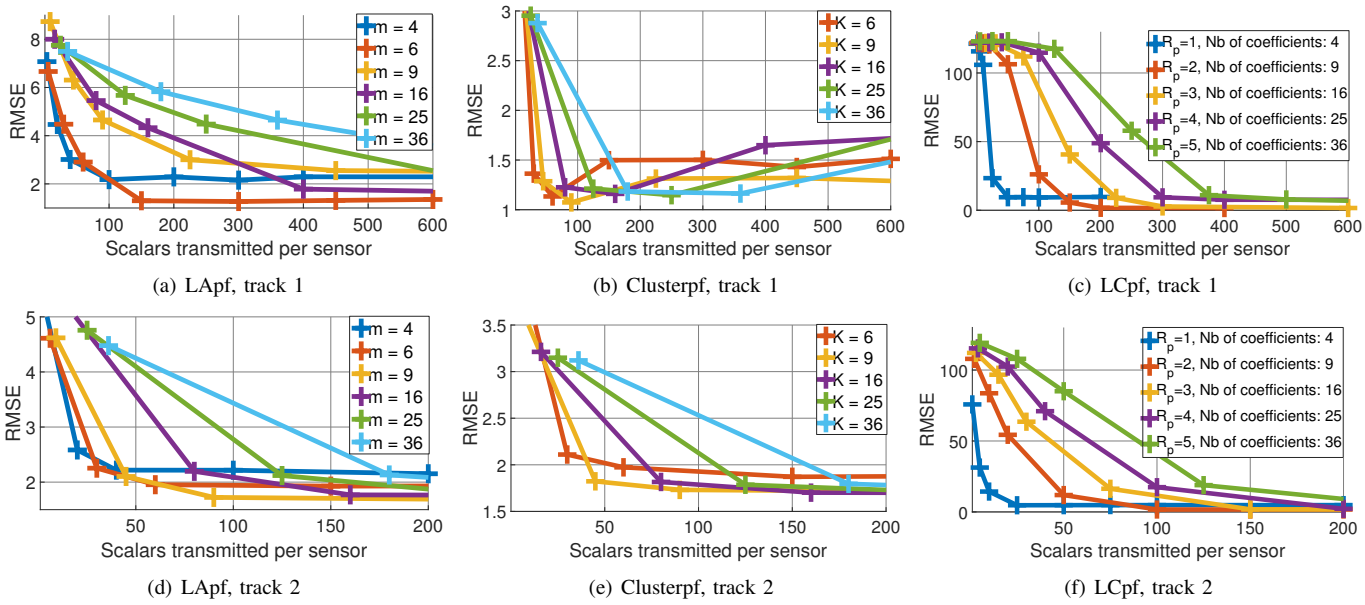
Consider next Clusterpf. We omit the curve for $K = 4$ since the performance is considerably worse (RMSE>5) than the other values. For track 1, the $K = 9$ curve gives a good trade-off between communication overhead and RMSE. For track 2, $K = 9$ gives low RMSE at low communication overhead and lowest RMSE at higher overhead.

Finally, consider LCpf. At low communication overhead (fewer than 50 scalars per sensor per iteration), $R_p = 1$ consistently yields the best performance by a significant margin for both tracks. We thus choose $R_p = 1$ for all LCpf simulations.

E. Comparison to other filters

Next we compare the tracking performance of the different distributed particle filters. Since the number of coefficients differs for each algorithm, we consider different values of NGossip and compare the RMSE in terms of the total number of scalars transmitted per sensor per time step.

CSSpf uses 6 basis functions by design [14]. For LCpf and LCpf-GS, the number of basis functions is 4 (given max polynomial degree R_p=1). As mentioned above, LApf retains 6 eigenvectors for track 1 and 9 eigenvectors for track 2, and Clusterpf uses 9 clusters. Finally, since the state vector is four-dimensional, GApf needs to transmit 14 scalars: 4 for the mean



(a) LApf, track 1 (b) Clusterpf, track 1 (c) LCpf, track 1

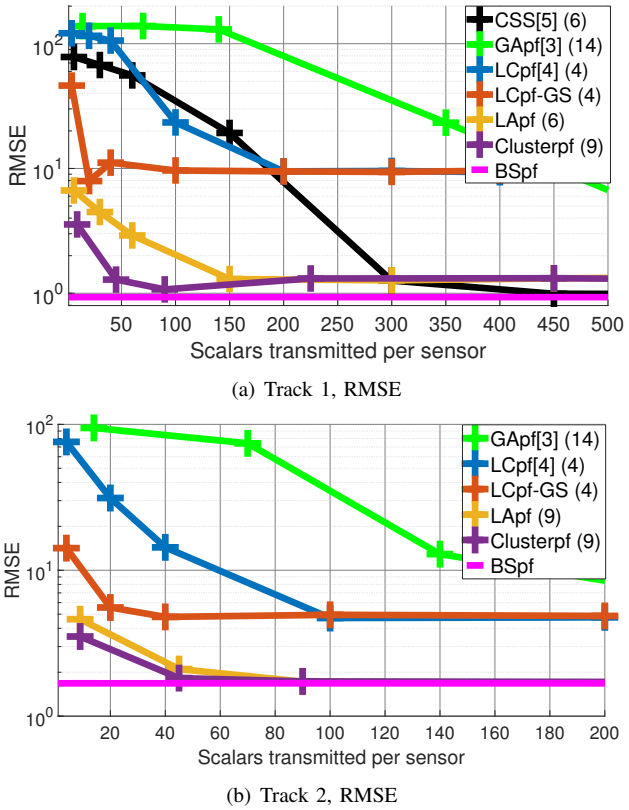
(d) LApf, track 2 (e) Clusterpf, track 2 (f) LCpf, track 2

Fig. 4. RMSE of LApf, Clusterpf and LCpf with respect to m (LApf), K (Clusterpf), R_p (LCpf) and NGossip. The x-axis represents the total number of scalars transmitted per sensor per time step. All results are averaged over 200 random trials.

vector and 10 for the upper triangular part of the covariance matrix.

Fig. 5 shows the simulation results for both tracks. Consider track 1 first. LApf and Clusterpf outperform the other distributed filters by a significant margin for overhead < 300. More interestingly, these two filters have adequate tracking performance even with just 1 gossip iteration per time step. In contrast, all other filters break down at low overhead and have very high RMSE ($>$10). The LCpf and GApf have the worst performance by far and do not come close to the performance of BSpf even with overhead exceeding 500. The CSSpf's RMSE drops sharply and is on-par with that of BSpf for overhead exceeding 300. Finally, the LCpf-GS outperforms LCpf by a large margin for overhead < 200. This confirms our previous conjecture that reducing the values of $||\Psi||_\infty||\alpha||_\infty$ can improve the performance of LCpf.

Consider the results for track 2. The LCpf and GApf break down at low communication overhead. For GApf, the RMSE drops below 10 after overhead exceeds 200. For LCpf, the RMSE reaches a plateau after overhead exceeds 100. The LCpf-GS again consistently outperforms LCpf at low overhead ($<$ 100). The LApf and Clusterpf consistently outperform the other fitlers by a large margin. For both proposed filters, an overhead of 80 scalars per sensor is sufficient to match the performance of BSpf. Furthermore, even with just one gossiping iteration, the LApf and Clusterpf are able to yield fairly robust tracking performance.



(a) Track 1, RMSE

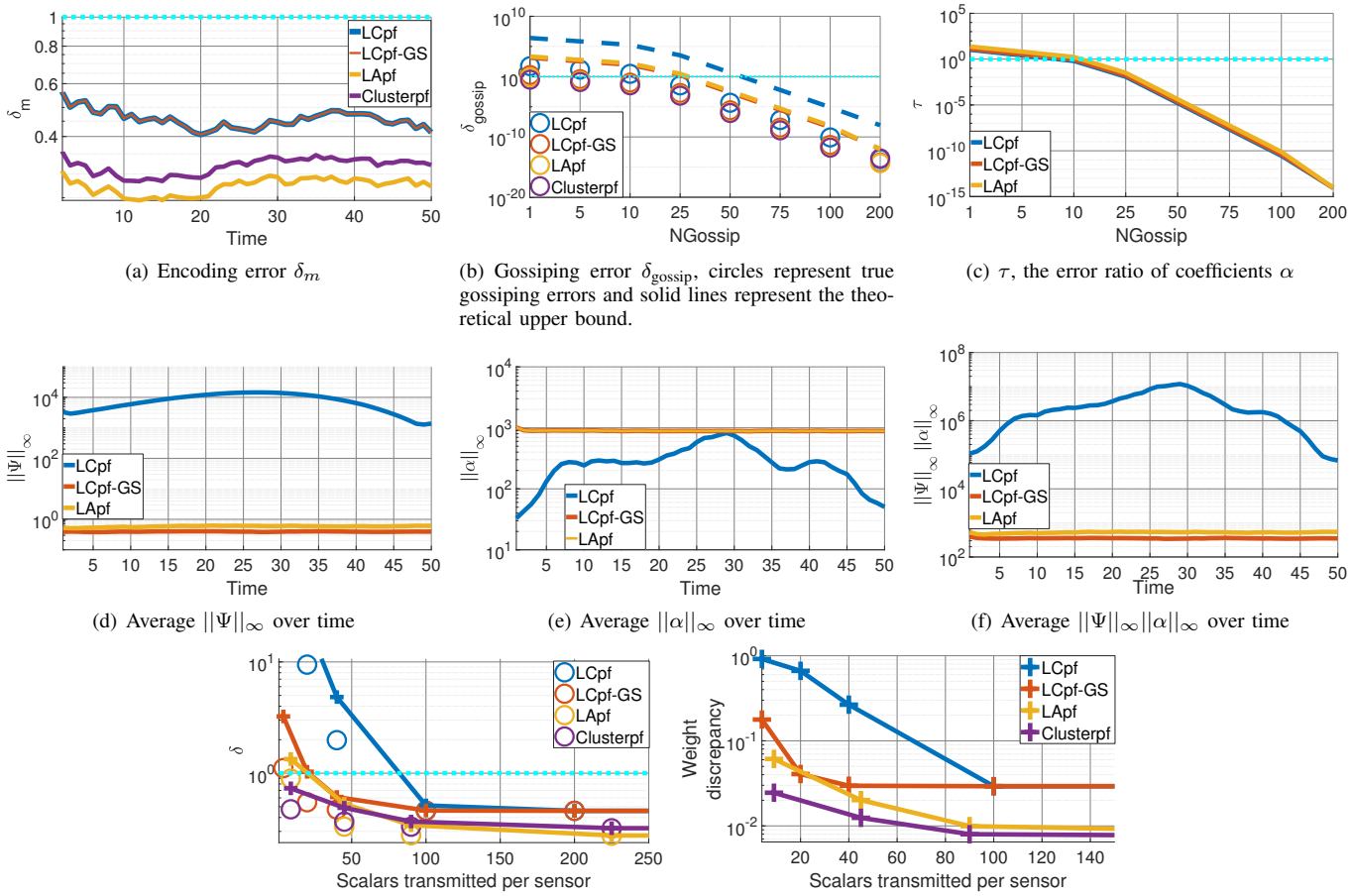
(b) Track 2, RMSE

Fig. 5. RMSE of all filters, averaged over 200 Monte Carlo trials, for test tracks 1 and 2. The numbers in brackets indicate the number of scalars transmitted per sensor per gossip iteration. The x-axis represents the total number of scalars transmitted per sensor per time step.

F. Error analysis

Section V presented an error bound for the proposed filters and showed that said bounds depend on the basis functions used to encode the particle log-likelihoods. To validate the derived error bounds, we run a centralized bootstrap filter on both tracks and at each time step re-compute the particle weights using the distributed filters. We compute and report the following values for CSSpf, LCpf, LCpf-GS, LApf and Clusterpf:

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}



(a) Encoding error δ_m

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

(c) τ, the error ratio of coefficients α

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

(e) Average $||\alpha||_\infty$ over time

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

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

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

Fig. 6. Distributed particle filter error bounds with respect to $NGossip$ and time for track 2. $N = 1000$, $R_p = 1, m = 9, K = 9$. The cyan dotted line denotes the $y = 1$ threshold value. (Best viewed in color.)

3) $\max_{\alpha_j} \frac{|\hat{\alpha}_j - \alpha_j|}{|\alpha_j|}$: estimate of τ
4) $\frac{\tau||\Psi||_\infty||\alpha||_\infty}{\min_{x_i}|\gamma(x_j)(1-\delta_m)|}$: 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

We emphasize that the overall error bound $(1+\delta_m)\delta_{\text{gossip}} + \delta_m$ applies to Clusterpf. However, since Clusterpf does not use a linear transformation, the upper bound for δ_{gossip} does not apply to Clusterpf and thus we do not plot the corresponding curves for Clusterpf. We run the same experiments for both tracks and observe similar trends and phenomenon in both cases, so we only show the results for track 2 in Fig. 6.

Consider first the encoding error δ_m. The LApf has the lowest encoding error followed by Clusterpf. The LCpf and LCpf-GS have the highest encoding error. Note that the encoding error values (and by extension the ranking of the filters) can be reduced by simply increasing the number of coefficients. Therefore we are more interested in ensuring that the error remains bounded, which is indeed the case for all four filters. 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 remain constant over different values of NGossip.

Consider next the gossiping error. For LApf, Clusterpf and LCpf-GS, the gossiping error drops below 1 for NGossip$\geq$5; although the error for all three filters is fairly close to 1 even at NGossip=1. For LCpf, δ_{gossip} falls below 1 only when NGossip $\geq$ 25. We also plot the derived upper bounds. While these bounds are quite loose, their overall trends are consistent with the actual values.

Fig. 6(c) shows the average value of $\tau = \max_j |\hat{\alpha}_j - \alpha_j|/|\alpha_j|$. The τ values decrease with more gossip iterations as expected. For all filters, the τ values are very close. This is expected since τ is independent of the choice of the filters.

Figs. 6(d)-(f) show the average values of $||\Psi||_\infty$, $||\alpha||_\infty$ and $||\Psi||_\infty||\alpha||_\infty$ over time for LCpf, LCpf-GS, and LApf. Consider $||\Psi||_\infty$ first. The LCpf has the largest curve by several orders of magnitude. For LApf and LCpf-GS, the value is very small as expected given the orthonormal transformation matrix. For $||\alpha||_\infty$, the LCpf has the lowest curve and the value fluctuates over time. In contrast, the LCpf-GS and LApf curves have nearly constant values. Finally, when we consider the product term $||\Psi||_\infty||\alpha||_\infty$, the LCpf has the highest curve. Conversely, the LCpf-GS has the lowest curve followed closely

by LApf. These results, combined with Eq. (22), suggest that, for a given NGossip (and by extension τ), LCpf would have the highest gossiping error bound followed by LCpf-GS and LApf. While a higher error bound does not necessarily equate to higher gossiping error (as the bound can be quite loose), the comparison of these bounds can still provide insights into the potential performance of the filters. 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. 6(g) shows the overall error δ with respect to scalars transmitted per sensor per time step. For LApf and Clusterpf, $\delta < 1$ for all values of overhead. For LCpf-GS, δ drops below 1 for overhead > 20. For LCpf, the error drops below 1 for overhead exceeding 100. For all filters, the upper bound $(1 + \delta_m)\delta_{\text{gossip}} + \delta_m$ is fairly close to the true value.

Finally, Fig. 6(h) shows the average discrepancy in normalized particle weights. The LCpf curves is significantly higher than the other curves until overhead > 100. The Clusterpf has the lowest weight discrepancy followed by LApf. We also note that there is a clear correlation between the weight discrepancy and the error δ (and consequently the overall tracking performance).

IX. 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. Based on these insights, we also present a modified likelihood consensus particle filter in which we construct a set of orthonormal basis functions. The modified filter is shown to outperform the original likelihood consensus filter by a large margin. For future work, we may consider filter implementations in which parameters like m or K are automatically adapted by the network without any fine-tuning from the users.

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," in *Proc. IEEE/ACM Info. Processing in Sensor Networks*, Berkeley, CA, USA, Apr. 2004, pp. 99–107.

[3] B. N. Oreshkin and M. Coates, "Asynchronous distributed particle filter via decentralized evaluation of Gaussian products," in *13th Conf. Inform. Fusion*, Edinburgh, UK, Jul. 2010, pp. 1–8.

[4] 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.

[5] 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.

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

[7] 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.

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

[9] 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.

[10] S. Farahmand, S. I. Roumeliotis, and G. B. Giannakis, "Set-membership constrained particle filter: Distributed adaptation for sensor networks," *IEEE Trans. Signal Processing*, vol. 59, pp. 4122–4138, Jun. 2011.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

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

[17] N. G. M. S. Arulampalam, S. Maskell and T. Clapp, "A tutorial on particle filters for online nonlinear/non-Gausssian Bayesian tracking," *IEEE Trans. Signal Process.*, pp. 174–188, Aug. 2002.

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

[19] 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.

[20] A. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, "Gossip algorithms for distributed signal processing," *Proceedings of the IEEE*, vol. 98, no. 11, 2010.

[21] 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.

[22] F. Iutzeler, P. Ciblat, and J. Jakubowicz, "Analysis of max-consensus algorithms in wireless channels," *IEEE Trans. Signal Processing*, vol. 60, no. 11, 2012.

[23] S. Datta 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.

[24] B. Mohar, "The Laplacian spectrum of graphs," in *Graph Theory, Combinatorics, and Applications*, Y. Alavi, G. Chartrand, O. R. Oellermann, and A. J. Schwenk, Eds. Wiley, 1991, vol. 2, pp. 871–898.

[25] 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.

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

[27] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Johns Hopkins University Press, 1996.

[28] A. V. Knyazev, "Toward the optimal preconditioned eigensolver: Locally optimal block pre-conditioned conjugate gradient method," *SIAM J. Scientific Computing*, no. 2, pp. 517–541.

[29] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Aug. 2007.

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

[31] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.

[32] P. Del Moral, *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Springer, 2004.

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

[34] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Applications to Tracking and Navigation*. Hoboken, NJ, USA: Wiley, 2004.

[35] D. T. Lee and A. K. Lin, "Generalized Delaunay triangulation for planar graphs," *Discrete & Computational Geometry*, vol. 1, pp. 201–217, Sep 1986.