

Distributed particle filter for bearing-only tracking

Jun Ye Yu

January 11, 2018

1 Introduction

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

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

2 Problem statement

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

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

In this report, we focus on bearing-only tracking. Each sensor receive a bearing measurement corrupted by additive zero-mean Gaussian noise, and has the following measurement model:

$$H_s(X) = \arctan 2 \left(\frac{x_t - x_s}{y_t - y_s} \right) + \eta_s \quad (1)$$

where $\eta_s \sim \mathcal{N}(0, \sigma_s)$ is the measurement noise.

3 Distributed particle filters for bearing-only tracking

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

requires the computation of joint log-likelihood:

$$\begin{aligned} w_i(k) \propto \log(f(z_1(k), \dots, z_S(k) | X_i(k))) &\propto \sum_{s=1}^S \frac{-(z_s - H_s(X))^2}{2\sigma_s^2} \\ &= \sum_{s=1}^S \frac{-(z_s)^2 - H_s(X)^2 + 2z_s H_s(X)}{2\sigma_s^2} \end{aligned} \quad (2)$$

where measurements from different sensors are assumed to be conditionally independent given the target state.

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

3.1 Constraint sufficient statistics particle filter

In the CSSpf, the likelihood function is approximated as follows:

$$\log(f(z_s | X) \approx \sum_{j=1}^6 G_{s,j} F_j(X) \quad (3)$$

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

The functions $F_j(X)$ depend only on X and are known to all sensors. The sufficient statistics $G_{s,j}$ depend only on local information from sensor s . In other words, we approximate the log-likelihood function by the combination of six basis functions $F_j(X)$ with corresponding coefficients $G_{s,j}$.

This formulation leads to the following approximate log-likelihood function

$$\log(f(z_1, \dots, z_S | X) \approx \sum_{j=1}^6 F_j(X) \left(\sum_{s=1}^S G_{s,j} \right) \quad (4)$$

where the summation terms $\left(\sum_{s=1}^S G_{s,j} \right)$ can be interpreted as the global sufficient statistics. These global sufficient statistics can be computed in a distributed manner by running six consensus algorithms in parallel. Note that the per-sensor communication overhead of CSSpf is constant since there are only six statistics to aggregate regardless of number of particles.

The CSSpf is derived based on LCpf and the six basis functions are specifically tailored for bearing-only tracking. For other measurement model, re-derivation of the filter is required.

3.2 Likelihood consensus particle filter

In LCpf, we approximate the measurement function as follows:

$$\hat{H}_s(X) = \sum_{j=1}^J \alpha_{s,j} \beta_j(X) \quad (5)$$

where $\beta_j(X)$ is the j^{th} sensor-independent basis function and $\alpha_{s,j}$ is the corresponding coefficient that encompasses all the local information of sensor s .

Plugging Eq. (??) into Eq. (??) yields

$$\begin{aligned} \log(f(z_1, \dots, z_S | X)) &\propto - \sum_{s=1}^S \frac{(z_s)^2}{2\sigma_s^2} - \sum_{s=1}^S \frac{\left(\sum_{j=1}^J \alpha_{s,j} \beta_j(X)\right)^2}{2\sigma_s^2} + \sum_{s=1}^S \frac{z_s \sum_{j=1}^J \alpha_{s,j} \beta_j(X)}{\sigma_s^2} \\ &= - \frac{\sum_{s=1}^S (z_s)^2}{2\sigma_s^2} - \sum_{j_1=1}^J \sum_{j_2=1}^J \frac{\sum_{s=1}^S \alpha_{s,j_1} \alpha_{s,j_2} \beta_{j_1}(X) \beta_{j_2}(X)}{2\sigma_s^2} + \sum_{j=1}^J \frac{\sum_{s=1}^S z_s \alpha_{s,j} \beta_j(X)}{\sigma_s^2} \\ &= - \frac{\sum_{s=1}^S (z_s)^2}{2\sigma_s^2} - \sum_{m=1}^M B_m(X) \left(\sum_{s=1}^S \frac{A_{s,m}}{2\sigma_s^2} \right) + \sum_{j=1}^J \beta_j(X) \left(\sum_{s=1}^S \frac{z_s \alpha_{s,j}}{\sigma_s^2} \right) \end{aligned} \quad (6)$$

where, for the last equality, we employ a suitable mapping $m \rightarrow (j_1, j_2)$, $M = J^2$, $B_m(X) = \beta_{j_1}(X) \beta_{j_2}(X)$ and $A_{s,m} = \alpha_{s,j_1} \alpha_{s,j_2}$.

Eq. (??) suggests that the joint log-likelihood can be constructed using $M + J$ consensus algorithms in parallel to compute the global sufficient statistics $\sum_{s=1}^S \frac{A_{s,m}}{2\sigma_s^2}$ and $\sum_{s=1}^S \frac{z_s \alpha_{s,j}}{\sigma_s^2}$. The first term in Eq. (??) is constant and independent of X and can thus be ignored.

We now describe how to compute the coefficients $\alpha_{s,j}$ using the least-square approach. Given the N particles X_i , we construct the $N \times J$ matrix Φ as follows:

$$\Phi = \begin{pmatrix} \beta_1(X_1) & \dots & \beta_J(X_1) \\ \dots & \dots & \dots \\ \beta_1(X_N) & \dots & \beta_J(X_N) \end{pmatrix} \quad (7)$$

For each sensor s , we also construct the following column vector

$$\Lambda_s = [H_s(X_1), \dots, H_s(X_N)]^T \quad (8)$$

where T denotes the transpose operation.

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

$$\alpha_s = (\Phi^T \Phi)^{-1} \Phi^T \Lambda_s \quad (9)$$

We note that the LCpf is not restricted to approximating the measurement function only. The same approach can be applied to estimate the particle log-likelihoods directly as in the case of CSSpf. The communication overhead per sensor depends directly on the number of coefficients which may in turn depend on the number of particles and other factors.

3.3 Laplacian approximation particle filter

In LAPf, we consider each particle X_i a vertex on a graph. The *Delaunay triangulation* (DT) on these particles is used to construct a graph. The resulting Laplacian matrix is used to construct a transformation that allows particle log-likelihoods to be represented using a minimal number of coefficients.

Let L denote the Laplacian matrix of the DT graph with eigenvalue decomposition $L = E^T \Lambda E$. The eigenvectors are used as the basis of transformation of particle log-likelihoods into Laplacian domain. Using

all N eigenvectors is obviously counterproductive since we incur the computational cost of eigendecomposition and achieve no reduction in communication overhead (i.e., we still have to aggregate N coefficients).

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

$$\alpha_s = E_m^T \gamma_s \quad (10)$$

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

$$\hat{\gamma} = E_m \alpha = E_m \sum_s \alpha_s \quad (11)$$

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

We note that LAPf is similar to CSSpf in that both filters encode the particle log-likelihoods directly using a minimal number of coefficients.

3.4 Clustering particle filter

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

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

In order to recover the individual particle joint log-likelihoods γ , we again construct the K -nearest-neighbor graph, compute the Laplacian matrix L , and then solve the following convex minimization problem:

$$\begin{aligned} & \underset{\gamma}{\text{minimize}} && \gamma^T L \gamma \\ & \text{subject to} && C \gamma = \gamma^c \end{aligned}$$

In other words, we seek to assign particle log-likelihood values that are smooth with respect to particle proximity while ensuring the aggregate particle values are equal to the cluster value.

4 Performance evaluation

4.1 Simulation setup

In this section, we evaluate and compare the performance of the four filters presented in Sec. ???. We also include a centralized bootstrap filter as baseline. We construct a network of $S = 9$ sensors in a square grid over a $75\text{km} \times 75\text{km}$ area and track a target traveling in counter-clockwise direction over 50 time steps. The sensors remain static over time. Fig. ?? shows the target trajectory and sensor positions.

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

$$X(k+1) = F(X(k)) + \xi(k) \quad (12)$$

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 $1 - P_{cv} = 0.95$.

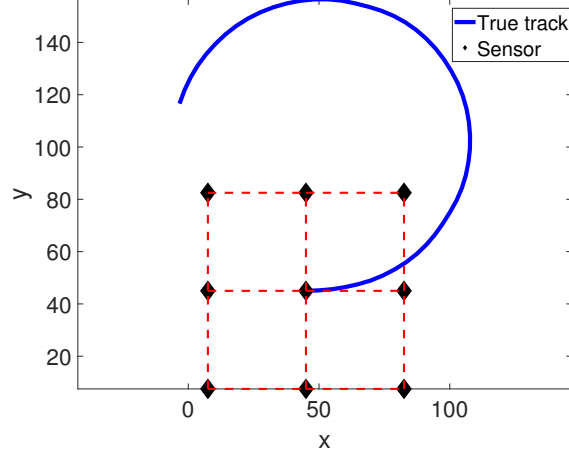


Figure 1: Target trajectory (blue curve) and sensor positions (black diamond). Sensors connected by red dashed lines are within broadcast range of each other.

For constant velocity, we have

$$F(X(k)) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

For coordinated turn, we have

$$F(X(k)) = \begin{bmatrix} 1 & 0 & \frac{\sin(\Omega(k))}{\Omega(k)} & -\frac{1-\cos(\Omega(k))}{\Omega(k)} \\ 0 & 1 & \frac{1-\cos(\Omega(k))}{\Omega(k)} & \frac{\sin(\Omega(k))}{\Omega(k)} \\ 0 & 0 & \cos(\Omega(k)) & -\sin(\Omega(k)) \\ 0 & 0 & \sin(\Omega(k)) & \cos(\Omega(k)) \end{bmatrix} \quad (14)$$

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

$$\Omega(k) = \frac{a}{\sqrt{\dot{x}^2(k) + \dot{y}^2(k)}} \quad (15)$$

with $a \in \mathbb{R}$ being the maneuver acceleration parameter.

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

$$H_s(X(k)) = \arctan 2 \left(\frac{x_t - x_s}{y_t - y_s} \right) + \eta(k) \quad (16)$$

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

$$Q = \sigma_a^2 \begin{bmatrix} \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 1 & 0 \\ 0 & \frac{1}{2} & 0 & 1 \end{bmatrix} \quad (17)$$

$$R = \sigma_\theta^2 \quad (18)$$

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

4.2 Algorithm setup

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

For LCPf, we use a set of basis functions involving all permutations of $x_t^i y_t^j$ with $0 \leq i, j \leq d$ where d is some user-specified max degree. For $d = 2$, the basis functions would be

$$\begin{aligned}\beta_1(X) &= x_t^0 y_t^0 = 1 \\ \beta_2(X) &= x_t^0 y_t^1 = y_t \\ &\dots \\ \beta_9(X) &= x_t^2 y_t^2\end{aligned}$$

Note that, due to our choice of basis functions, all particles must remain synchronized across all sensors as in the case of LAPf and Clusterpfp.

For LAPf, we construct a DT graph and retain $m < N$ eigenvectors as the basis of Laplacian transformation. For Clusterpfp, all particles are grouped into C clusters and a DT graph is constructed to recover individual particle weights.

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

$$G_i(t+1) = w_{ii}G_i(t) + \sum_{j \in N_i} w_{ij}G_j(t) \quad (19)$$

$$w_{ij} = \begin{cases} \frac{1}{1+\max(d_i, d_j)} & j \in N_i \\ 1 - \sum_{j \in N_i} w_{ij} & i = j \\ 0 & j \notin N_i \end{cases} \quad (20)$$

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

In our codes, we do not implement a loop for each gossip iteration. Instead we define an update matrix W where $W(i, j) = w_{ij}$. Therefore, given initial values $G(1) = [G_1(1), \dots, G_S(1)]^T$, the final values can be easily computed as

$$G(NGossip) = W^{NGossip}G(1) \quad (21)$$

Since the sensors remain static over time in our simulation, W remains static and a quick computation shows that $W_{final} = W^{31}$ is sufficiently close to an averaging matrix (i.e. $W_{final}(i, j) = 1/S$).

In the remainder of the section, we run a number of Monte Carlo simulations to evaluate the performance of individual algorithms. The track remains the same in each trial; but the measurements differ. We evaluate the algorithms' performances using three criterion: *root mean squared error* (RMSE) of position estimate, total runtime (from initialization to end of time step 50) and aggregate error ratio. The first two metrics are self-explanatory; so we will only explain the last one. Let G_{gossip} denote the vector of the approximate aggregate values computed using gossip and max consensus, and let G_{exact} denote the vector of exact aggregate values. We compute the ratio vector $|\frac{G_{gossip} - G_{exact}}{G_{exact}}|$ and report the average ratio. Ideally, for $NGossip$ approaching infinity, the ratio approaches 0 as the approximate aggregate value approaches the exact value.

4.3 Constraint sufficient statistics particle filter

Consider first CSSpf. Each sensor has a fixed communication overhead at each time step (i.e., broadcast $6NGossip$ scalars). We thus study the trade-off between communication overhead and tracking performance.

Fig. ?? shows the boxplots of RMSE, total runtime and aggregate error ratio with respect to number of gossip iterations and number of particles over 200 random trials. Each data point is averaged over all sensors and all time steps.

The RMSE is very high for $NGossip = 10$ which can be attributed to the gossip algorithms not converging in such few iterations and leading to high errors in the final aggregate values of the global sufficient statistics. For $NGossip \geq 20$ and $N \geq 100$, the RMSE is fairly constant and, at least for the given track, more gossip iterations and more particles do not yield significant improvement in tracking performance.

The runtime does not increase with higher $NGossip$ due to our implementation of the gossip algorithms. When N increases, the runtime increases as expected.

Finally, consider the aggregate error ratio. With increasing $NGossip$, the ratio decreases exponentially (note the log-scale of the Y-axis). For $NGossip = 10$, the gossip algorithms clearly have not converged and the error ratio can be as high as 10%. This explains the high RMSE values. For $NGossip = 20$, the ratio drops close to 0.02 which is low enough to yield adequate tracking performance. For higher values of $NGossip$, the ratio drops even further; although the tracking performance improvement is marginal at best. As N increases, the aggregate error ratio does not change significantly. This is to be expected since the number of sufficient statistics to be aggregated is constant and independent of the number of particles.

4.4 Likelihood consensus particle filter

The max degree d offers a trade-off between tracking performance and computational/communication overhead. Higher degree d generates more basis functions and should yield better approximation of the measurement model. On the other hand, more basis functions lead to more computation, more communication and longer runtime. In fact, the total number of basis functions grows exponentially with d (i.e., $O(d^2)$).

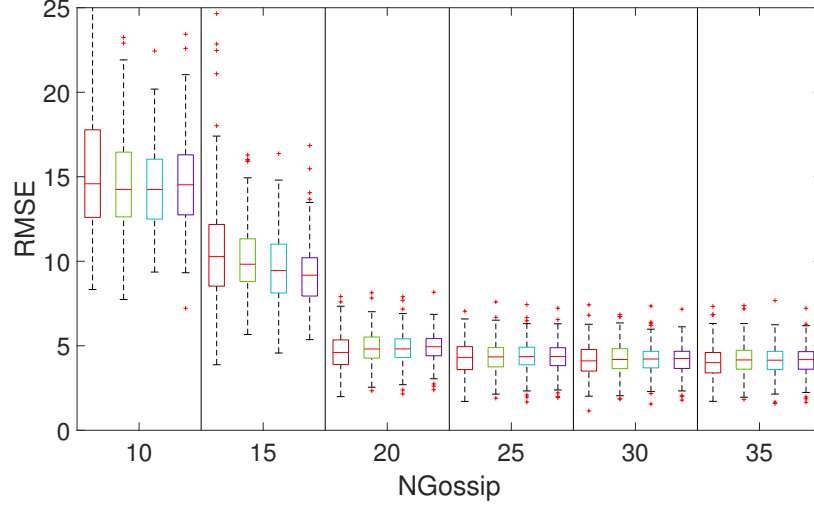
Fig. ?? shows the boxplots of RMSE and total runtime with respect to d . Note that, for this particular set of trials, all aggregates are computed exactly without gossiping. The runtime increases for higher d due to the additional computational overhead as expected. For all values of d , the RMSE is fairly consistent which suggests that additional basis functions from $d \geq 1$ do not improve the approximation of the measurement model $H(X_i)$ by any significant margin. To confirm our conjecture, we compute the following metric. Let $H_{true}(X_i)$ denote the true expected measurement of particle X_i and let $H_{approx}(X_i)$ denote the approximate value computed using the basis functions. We then compute the measurement model approximation error $|H_{exact}(X_i) - H_{approx}(X_i)|$ averaged over all particles, all sensors and all time steps. Fig. ?? shows the boxplot of average measurement model approximation error with respect to d . While the average error does fluctuate somewhat with respect to d , the error remains very small in all cases (less than 1 degree) compared to the standard deviation of measurement noise (5 degrees). These results suggest that $d = 1$ is sufficient to yield adequate tracking performance while minimizing computational overhead.

Fig. ?? shows the boxplots of RMSE, total runtime and aggregate error ratio with respect to number of gossip iterations and number of particles over 200 random trials for $d = 1$. Each data point is averaged over all time steps. For LCpf, a minimal of 40 gossip iterations are necessary to ensure adequate tracking performance. Furthermore, recall that, at $d = 1$, a total of 20 scalars needs to be aggregated across the network. In other words, the communication overhead of LCpf is 6 times that of CSSpf (thrice the scalars to aggregate and twice as many gossip iterations) to achieve similar tracking performance. If we compare Fig. 2(c) and Fig. 5(c), we note that CSSpf and LCpf exhibit very similar level of aggregate error ratios for the same $NGossip$. This suggests that LCpf is significantly more susceptible to error induced by distributed aggregation. On the other hand, LCpf requires only half the runtime of CSSpf.

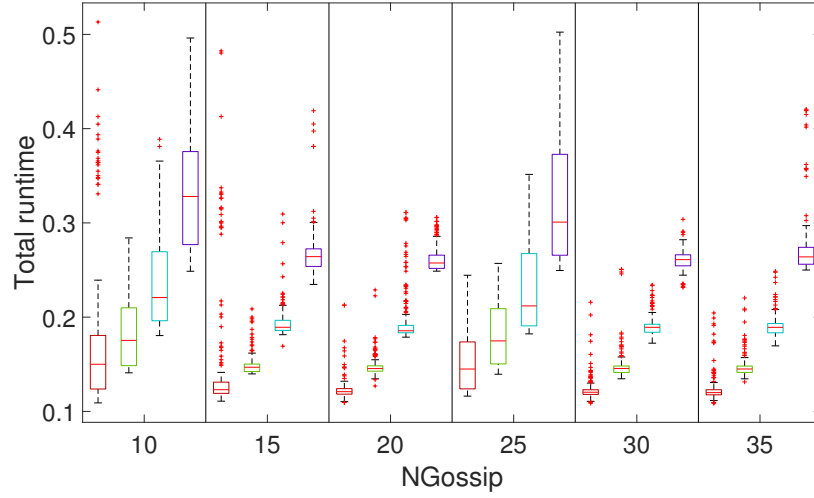
4.5 Laplacian approximation particle filter

The LAPf has one parameter of interest: m , the number of eigenvectors to encode the particle log-likelihoods. Fig. ?? shows the boxplots of RMSE, runtime and aggregate error ratio of LAPf with respect to m and $NGossip$ for $N = 500$.

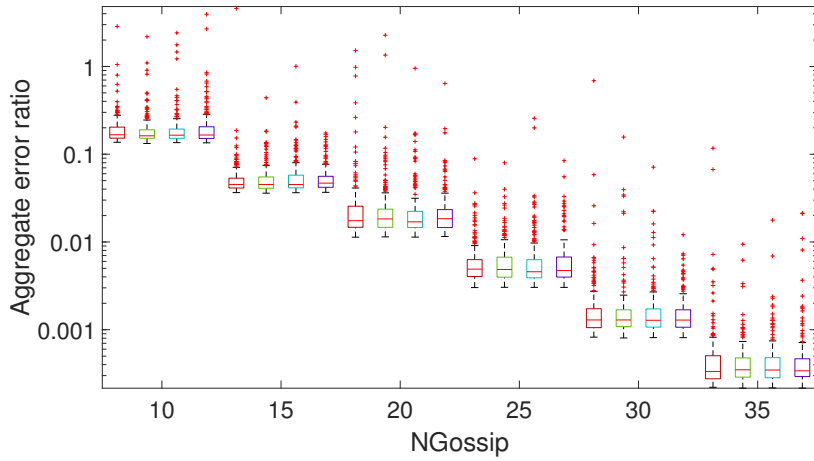
The RMSE is fairly stable for all values of $NGossip$ and m . This suggests two things. First, 6 Eigenvalues are sufficient to encode most information regarding the particle log-likelihood. Second, as few as 10



(a) RMSE

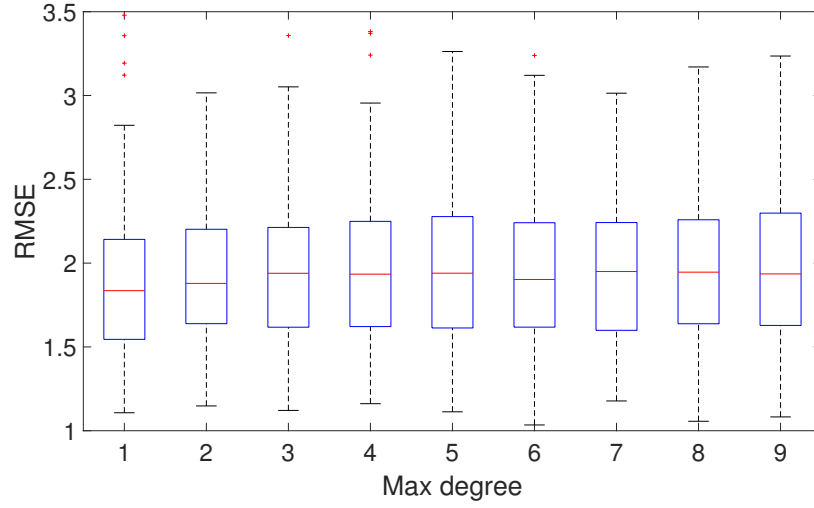


(b) total runtime

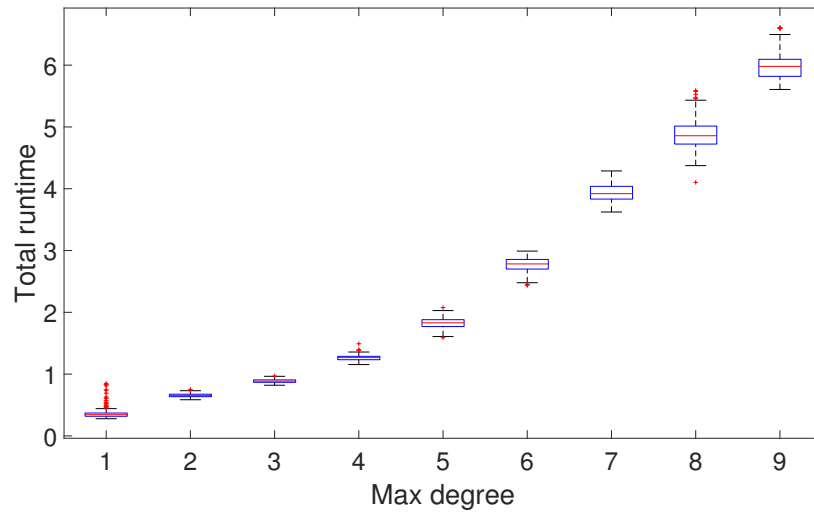


(c) total runtime

Figure 2: Boxplot of RMSE, runtime and aggregate error ratio of CSSpf with respect to $NGossip$ and N for 200 Monte Carlo trials. For each value of $NGossip$, from left to right, $N = 100, 250, 500, 1000$



(a) RMSE



(b) total runtime

Figure 3: Boxplot of RMSE and total runtime of LCpf with respect to d . All aggregations are computed exactly without gossip.

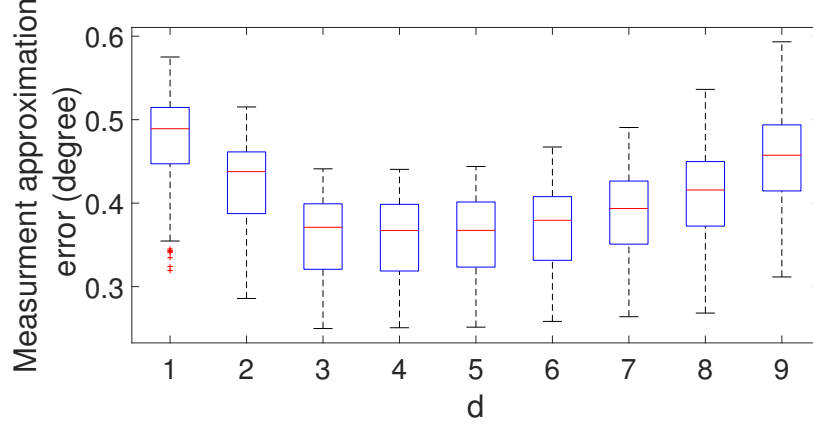


Figure 4: Boxplot of average measurement model approximation error with respect to d

gossip iterations per time step are sufficient to yield robust tracking performance. This strikes us as rather surprisingly since the average aggregate error of LAPf is actually higher than that of LCpf and CSSpf for the same values of $NGossip$. This would suggest that LAPf is surprisingly robust to error induced in distributed aggregation. Finally, the runtime is considerably higher than that of CSSpf and LCpf. The eigenvalue decomposition accounts for the bulk of the computational overhead which is independent of the values of $NGossip$ and m .

To confirm our conjecture regarding the choice of m , we compute the following metrics. Let \hat{w}_i denote the approximate weight of particle X_i computed using the Laplacian transformations and let w_i denote the true particle weight. We compute and report the discrepancy $|\hat{w}_i - w_i|$ averaged over all particles and all time steps. Fig. ?? shows the boxplot of the weight discrepancy with respect to m . At $m = 6$, the average discrepancy is as low as $4e - 4$. Increasing m to 100 only reduces the error by a factor of 3 while increasing the communication overhead by more than 15 times.

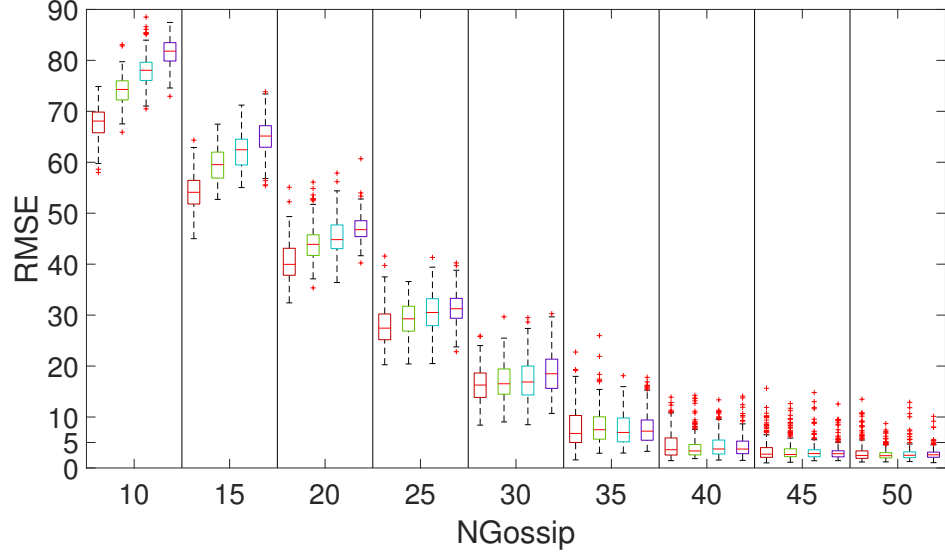
4.6 Cluster particle filter

The number of clusters C offers a trade-off between tracking performance and computational/communication overhead. Higher number of clusters should yield better performance albeit at higher overhead. In fact, one might argue that the parameter C is analogous to m in LAPf. Fig. ?? shows the results of 200 Monte Carlo trials. Similar to the results of LAPf, the RMSE is fairly constant for all considered values of $NGossip$ and C . The total runtime increases linearly with larger C because clustering the particles accounts for the majority of computational overhead and the workload is dependent on the number of clusters. The aggregate error ratio of Clusterpf appears on-par with those of CSSpf and LCpf and is lower than that of LAPf. Nonetheless, the robust tracking performance at $NGossip = 10$ suggests that Clusterpf is very robust to error induced in distributed aggregation.

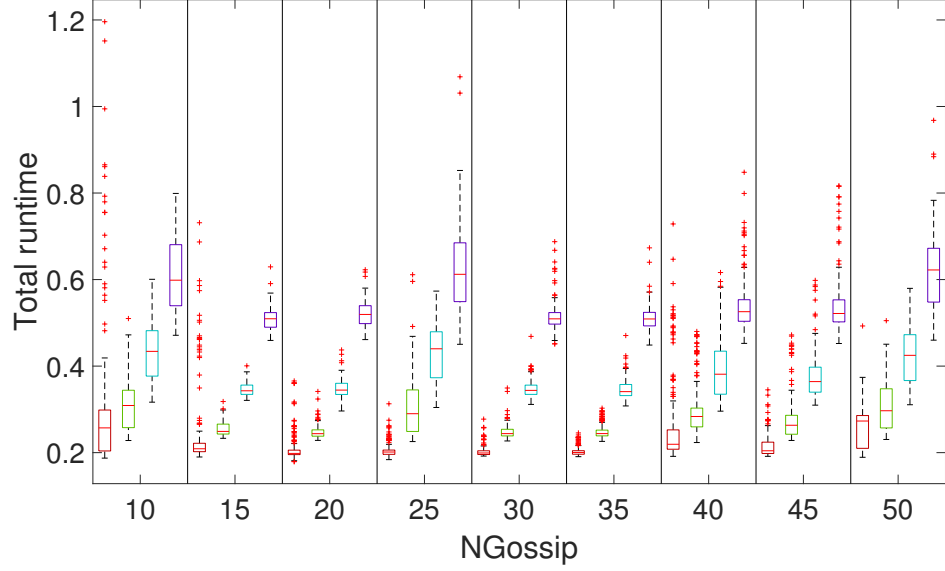
As in the case of LAPf, we compute and report the average weight discrepancy in Fig. ?. Increasing C from 6 to 100 decreases the approximation error by no more than half at the cost of a tenfold increase in communication overhead.

4.7 Performance comparison between filters

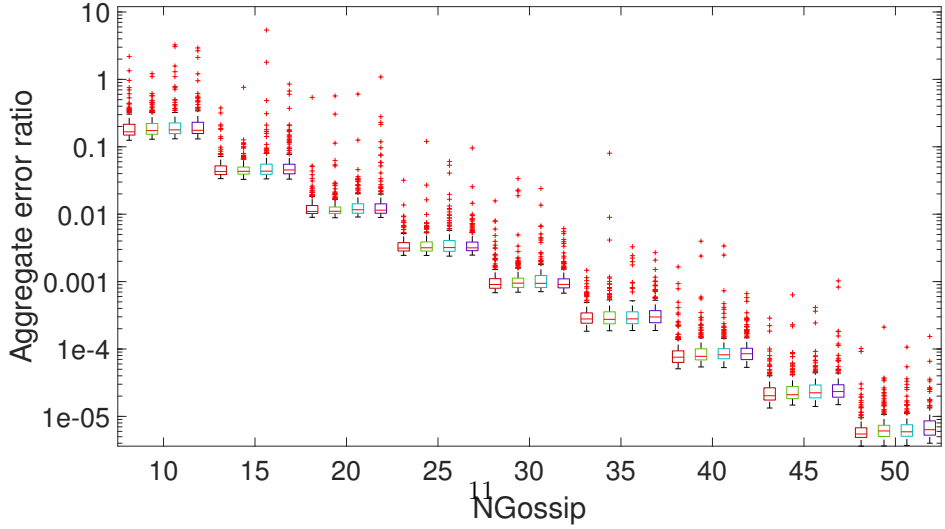
In this section, we compare the four filters against each other. We also include a centralized bootstrap particle filter as baseline. All four filters contain a number of algorithm-specific parameters that affect their performance, and tweaking each parameter for optimal performance is difficult. Instead we adjust the parameters such that all algorithms have the same communication overhead at each time step and compare their performance in terms of RMSE and total runtime.



(a) RMSE

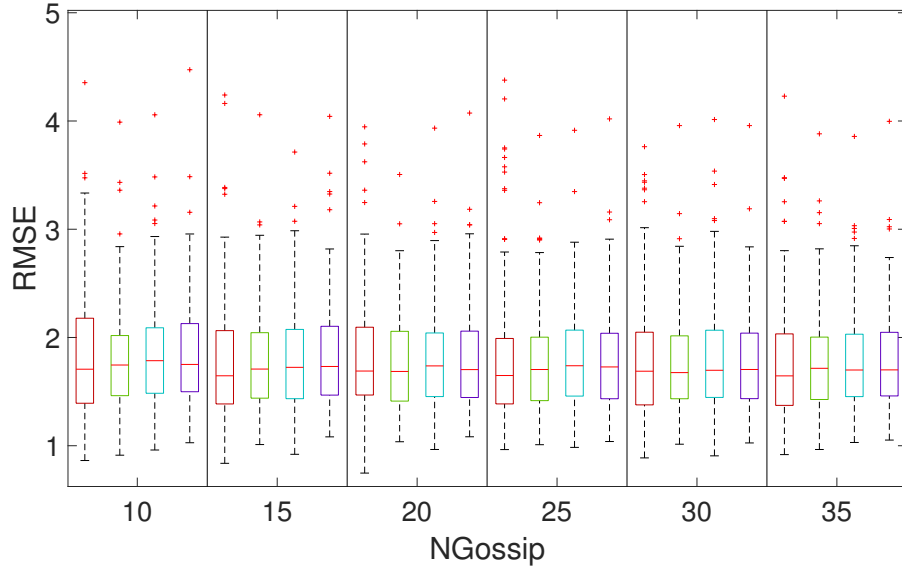


(b) total runtime

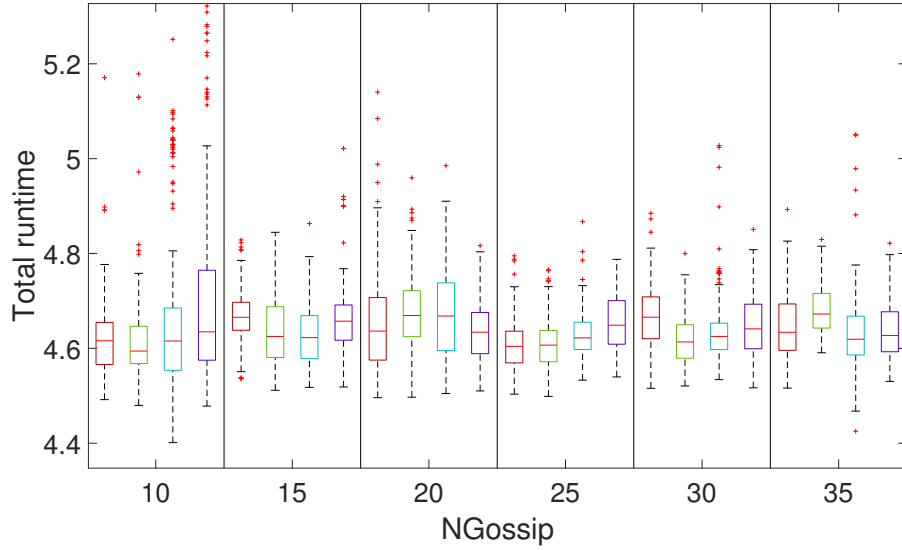


(c) total runtime

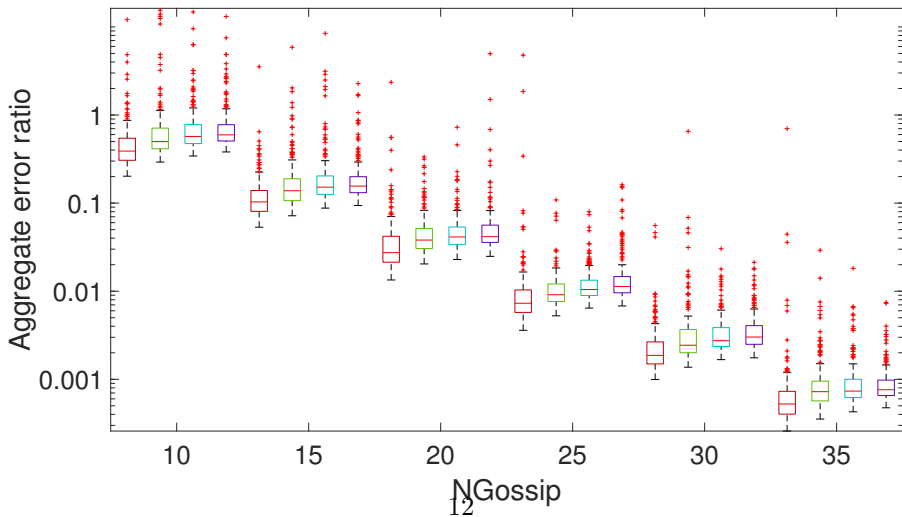
Figure 5: Boxplot of RMSE, runtime and aggregate error ratio of LCpf with respect to $NGossip$ and N for 200 Monte Carlo trials. For each value of $NGossip$, from left to right, $N = 100, 250, 500, 1000$



(a) RMSE



(b) total runtime



(c) total runtime

Figure 6: Boxplot of RMSE, runtime and aggregate error ratio of LAPf with respect to $NGossip$ and m for 200 Monte Carlo trials ($N = 500$). For each value of $NGossip$, from left to right, $m = 6, 20, 50, 100$

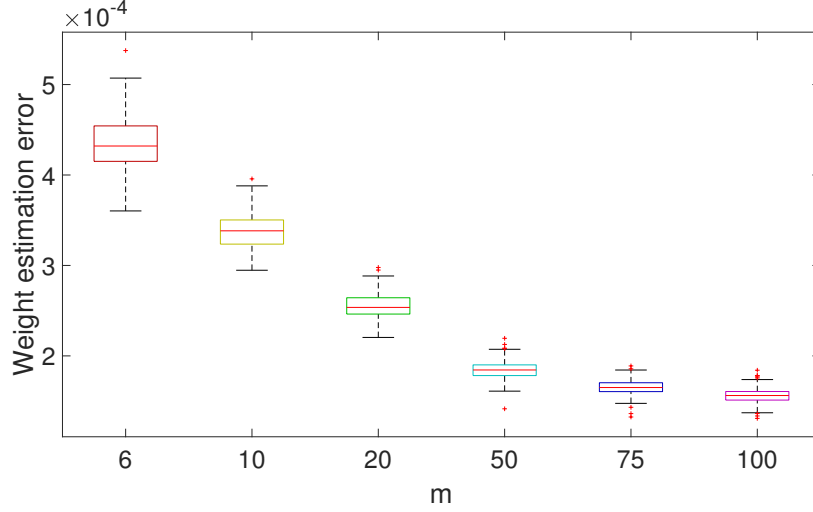


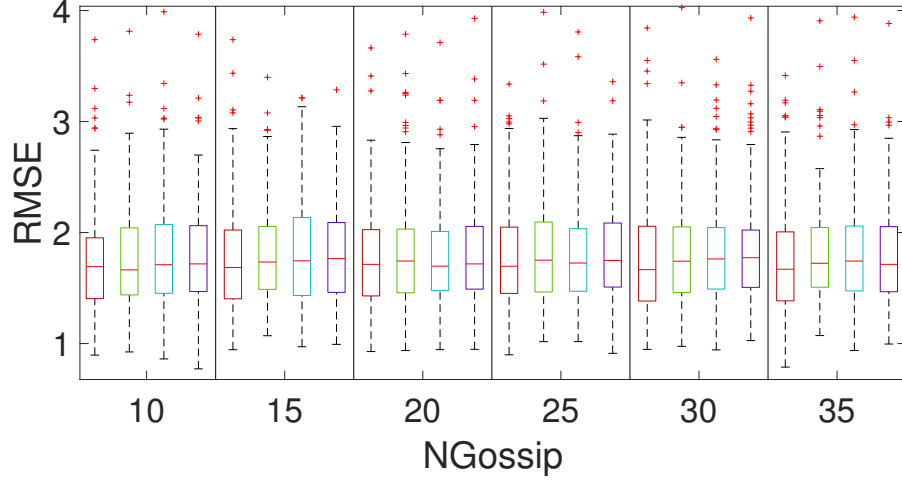
Figure 7: Boxplot of weight discrepancy with respect to m

Fig. ?? shows the first test track. Fig. ?? shows the average RMSE and total runtime with respect to communication overhead per time step for all algorithms. For CSSpf and LCpf, we vary only the number of gossip iterations per time step. We run two variations of LAPf and Clusterpf. In the first version, we fix m and C , the number of scalars to aggregate per time step, and vary $NGossip$. In the second version, we fix $NGossip$ and vary m and C accordingly.

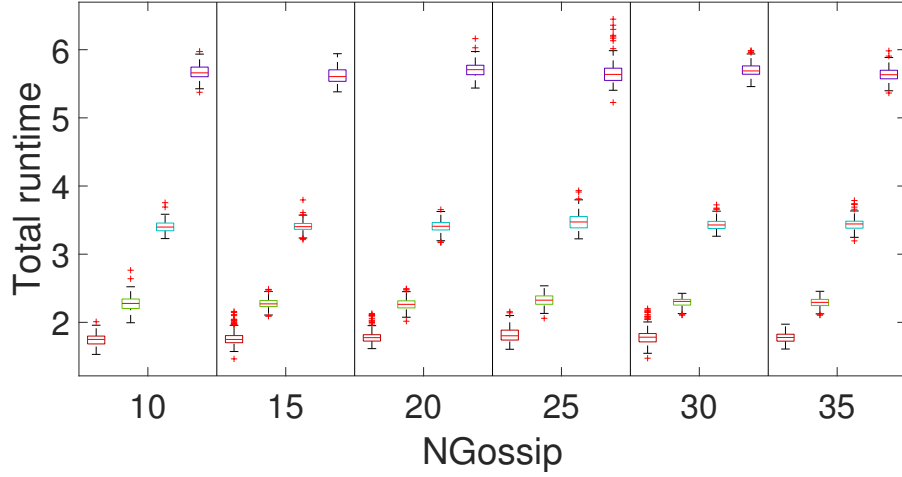
5 Conclusion

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

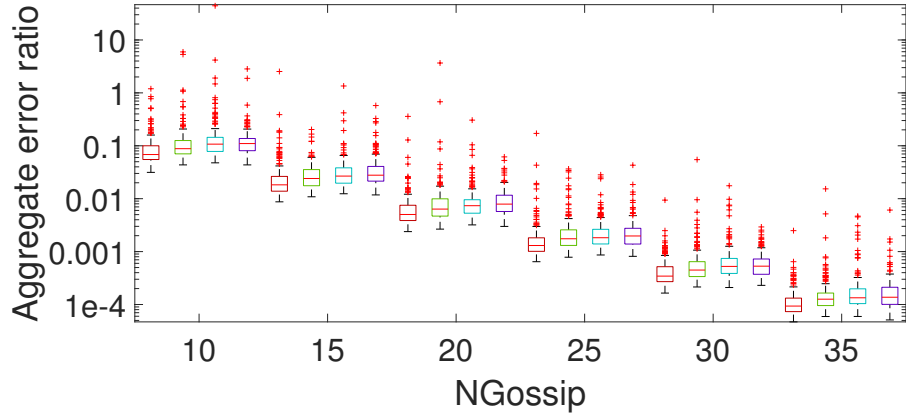
We study each individual algorithm’s performance and compare them against each other. CSSpf has the worst tracking performance by far. LCpf yields competitive tracking performance on-par with the centralized bootstrap particle filter and has very low runtime. On the other hand, it also has the highest communication overhead. LAPf has very high runtime due to the computational overhead of eigenvalue decomposition which renders it impractical for applications with strict timing constraints. Finally, Clusterpf has the second highest runtime but also yields competitive tracking performance.



(a) RMSE



(b) total runtime



(c) total runtime

Figure 8: Boxplot of RMSE, runtime and aggregate error ratio of Clusterpf with respect to $NGossip$ and C for 200 Monte Carlo trials ($N = 500$). For each value of $NGossip$, from left to right, $C = 6, 20, 50, 100$

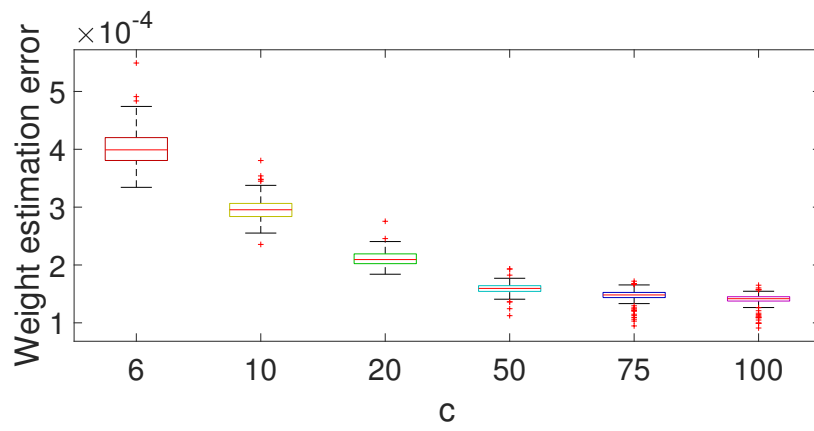


Figure 9: Boxplot of weight discrepancy with respect to C