Distributed particle filter for bearing-only tracking

Jun Ye Yu

December 12, 2017

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** [1], **LCpf** [2], **LApf** [3] and **Clusterpf** [4] respectively. We also include the centralized *bootstrap particle filter* (**BSpf**) as baseline.

The remainder of the report is organized as follows. Sec. 2 defines the tracking problem. Sec. 3 presents the particle filters. Sec. 3.4 compares the filters' performance and Sec. 5 concludes the report.

2 Problem statement

A network of S sensors collaboratively track a single moving target over time. The sensors have fixed position $[x_s, y_s], s = 1...S$. 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:

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

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:

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

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 $\gamma_s = [\log(f(z_s|X_1), ... \log(f(z_s|x_N))]^T$ denote the column vector containing the log-likelihoods of all N particles at sensor s. Similarly, let $\gamma = [\log(f(z_1, ..., z_S|X_1), ... \log(f(z_1, ..., z_S|x_N))]^T$ 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:

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

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

$$\begin{aligned}
F_1(X) &= 1 \\
F_2(X) &= x_t^2 \\
F_3(X) &= y_t^2 \\
F_4(X) &= -2x_t y_t \\
F_5(X) &= 2x_t \\
F_6(X) &= -2y_t
\end{aligned}$$

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

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

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:

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

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. (5) into Eq. (2) yields

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

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

Eq. (6) 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. (6) 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:

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

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

$$\Lambda_s = [H_s(X_1), ...H_s(X_N)]^T \tag{8}$$

where T denotes the transpose operation.

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

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

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. On the other hand, the communication overhead per sensor is not constant and depends on the number of coefficients.

3.3 Laplacian approximation particle filter

In LApf, we consider each particle X_i a vertex on a graph. The Euclidean distance between the particles is used to construct a K-nearest-neighbor 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 K-nearest-neighbor 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:

$$\alpha_s = E_m^T \gamma_s \tag{10}$$

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:

$$\hat{\gamma} = E_m \alpha = E_m \sum_s \alpha_s \tag{11}$$

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 [2]. 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:

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

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. 3. We also include a centralized bootstrap filter as baseline. We construct a network of $S = 9$ sensors in a square grid over a 75km $\times$ 75km area and track a target traveling in counter-clockwise direction over 50 time steps. The sensors remain static over time. Fig. 1 shows the target trajectory and sensor positions.

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

$$X(k + 1) = F(X(k)) + \xi(k) \tag{12}$$

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 $1 - P_{cv} = 0.95$.

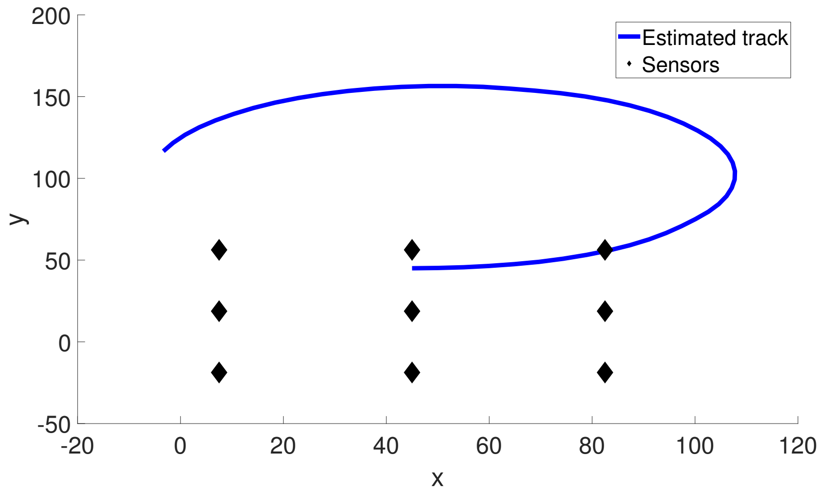


Figure 1: Target trajectory (blue curve) and sensor positions (black diamond)

For constant velocity, we have

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

For coordinated turn, we have

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

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

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

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

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

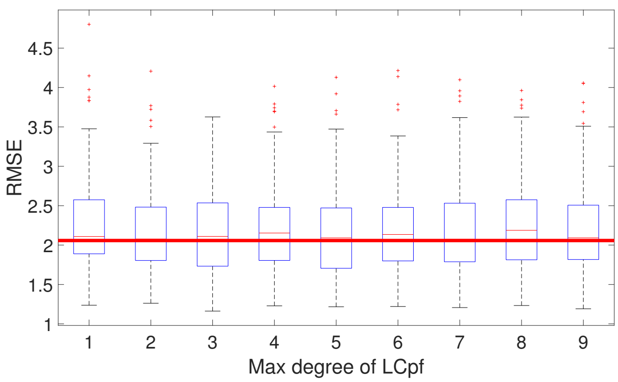
$$H_s(X(k)) = \arctan 2\left(\frac{x_t - x_s}{y_t - y_s}\right) + \eta(k) \tag{16}$$

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

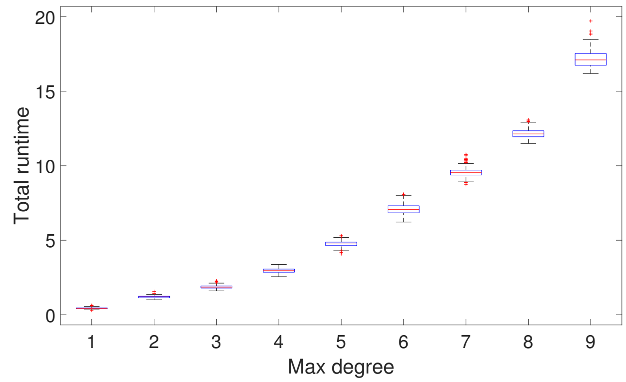
$$Q = \sigma_a^2 \begin{bmatrix} \frac{1}{3} & 0 & \frac{1}{2} & 0 \\ 0 & \frac{1}{3} & 0 & \frac{1}{2} \\ \frac{1}{2} & 0 & 1 & 0 \\ 0 & \frac{1}{2} & 0 & 1 \end{bmatrix} \tag{17}$$

$$R = \sigma_\theta^2 \tag{18}$$

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



(a) RMSE



(b) total runtime

Figure 2: Boxplot of RMSE and runtime of LCpf with respect to max degree for 100 Monte Carlo trials. The red solid line in the subfig a is the average RMSE of BSpf for comparison.

4.2 Algorithm setup

All particle filters use a total of $N = 1000$ 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 Clusterpf.

For LApf, we construct a K-nearest neighbor graph and retain $m < N$ Eigenvectors as the basis of Laplacian transformation. For Clusterpf, all particles are grouped into C clusters and a K-nearest neighbor 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. All summations are computed exactly without using any distributed algorithms. This ensures that no performance degradation is caused by consensus.

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 two criterion: *root mean squared error* (RMSE) of position estimate and total runtime (from initialization to end of time step 50).

4.3 Likelihood consensus particle filter

Consider first LCpf. The max degree d should offer a trade-off between tracking performance and total runtime. 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 and longer runtime. In fact, the total number of basis functions grows exponentially with d (i.e., $(d + 1)^2$).

Fig. 2 shows the boxplot of time-averaged RMSE and total runtime over 100 Monte Carlo trials. The total runtime increases with larger max degree as expected. On the other hand, we do not see any significant

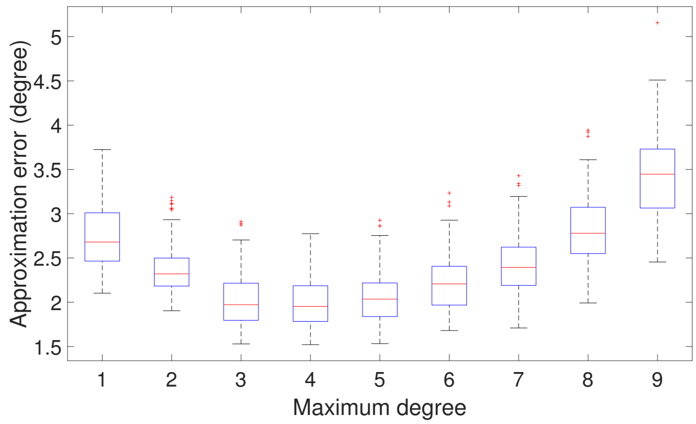


Figure 3: Boxplot of measurement approximation error with respect to max degree. All results are computed over 100 Monte Carlo trials

difference in RMSE for $1 \leq d \leq 9$. This suggests that the additional basis functions for $d > 1$ do not improve the approximation of the measurement model $H(X_i)$ by any significant margin.

To verify our conjecture, we compute the following metric. At each time step, we compute the aggregate approximation error over all S measurements for each particle $(\sum_{s=1}^{S} H_s(X_i) - \hat{H}_s(X_i))$, and report the average value $\frac{1}{N}\sum_{i=1}^{N}\sum_{s=1}^{S}(H_s(X_i) - \hat{(H)}_s(X_i))$.

Fig. 4.3 shows that the approximation error with respect to max degree. The approximation error decreases until $d = 4$ and increases afterwards; but the difference is rather small with a maximum difference of 1.5 degree (versus the 5 degrees standard deviation of measurement noise). This small difference in approximation error directly translates to negligible difference in tracking performance.

4.3.1 Laplacian approximation particle filter

The LApf has two parameters of interest: K, the number of nearest neighbors for the particle graph, and m, the number of Eigenvectors for the transformation. Different values of K yield different graphs over the same particles and by extension different Eigenvectors. Retaining more Eigenvectors yields a better approximation of particle log-likelihoods at the cost of higher computation overhead.

Table. 1 shows the average RMSE of LApf with respect to K and m. As m increases, the RMSE tends to decrease as expected since the approximation error is reduced. At $m = 1000$, there is no approximation error and the RMSE is constant for all values K and on-par with that of centralized BSpf. For $m = 6, 10, 20$, we see a decrease in RMSE as K goes up. For $m = 100, 200, 500$, the RMSE has small fluctuations over K. To get a closer look at the results, Fig. 4 shows the boxplot of RMSE for select values of m. For small value of m, we do see a steady decrease in average RMSE with increasing K; although the results can vary greatly from one trial to the nxt. For larger m, the RMSE is fairly consistent across K and independent MC trials.

To understand these trends, we compute and report at each time step the approximation error of normalized particle weights: $\sqrt{\sum_{i=1}^{N}(w_i - \hat{w}_i)^2}$ where w_i is the exact particle weight and $\hat{w}_i$ is the approximate value using m eigenvectors. Table. 2 shows the results. As expected, at $m = 1000$, there is no approximation error as all eigenvectors are used. For small values of m, we see a drastic decrease in approximation error as K increases. This in turn translates to lower RMSE. For $m \geq 100$, the average approximation error actually goes up with as K increases. We observe the same trends in Fig. 5. However, since the error remains small in case of high m, there is no significant impact on the overall tracking performance.

Table. 3 shows the total runtime of LApf with respect to K and m. For $1 \leq K \leq 10$, when m increases, the particle log-likelihoods are encoded using more coefficients. However, since this is a matrix multiplication operation, the difference in computational overhead is negligible and there is no significant variation in computational overhead. For $K = 20, 50, 100$, surprisingly, there is a significant fluctuation in runtime as

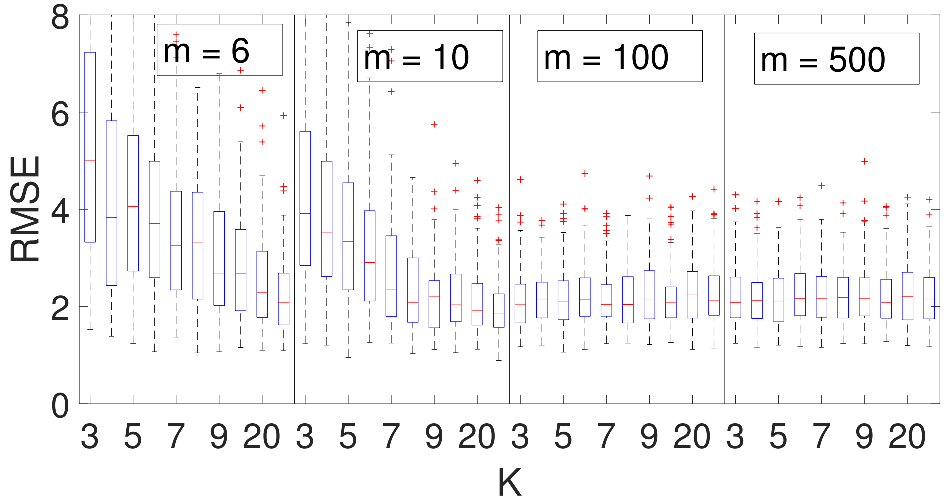


Figure 4: Boxplot of LApf RMSE for select values of m. The values of K are $3, 4, ...10, 20, 50$. All results are computed over 100 trials

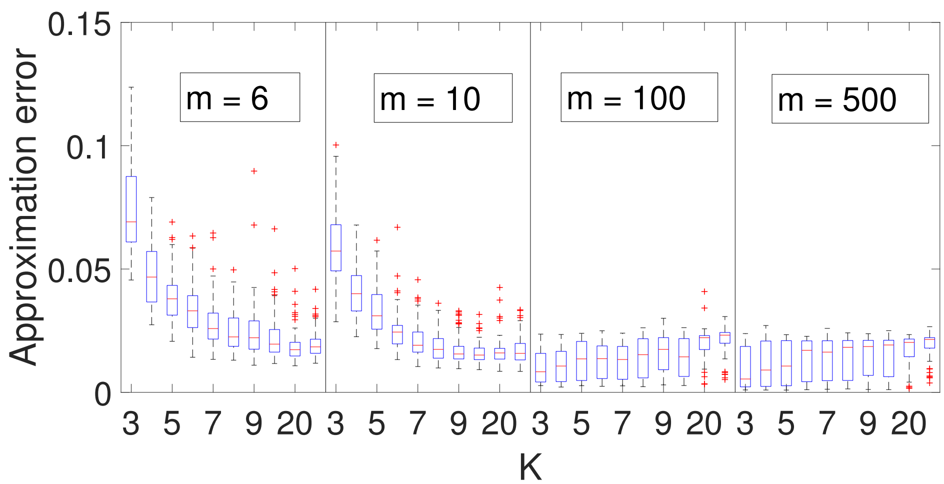


Figure 5: Boxplot of measurement approximation error for select values of m. The values of K are $3, 4, ...10, 20, 50$. All results are computed over 100 trials

	m = 6	10	20	100	200	500	1000
K = 3	5.4804	4.4057	3.9159	2.1505	2.1612	2.2161	2.1646
4	4.2176	3.8376	3.6131	2.2138	2.195	2.2501	2.1646
5	4.3284	3.6929	2.914	2.1928	2.2302	2.2292	2.1646
6	4.1016	3.3112	2.3407	2.2309	2.233	2.2877	2.1646
7	3.6605	2.8502	2.2711	2.1899	2.229	2.2864	2.1646
8	3.436	2.3306	2.1202	2.1923	2.235	2.2655	2.1646
9	3.1273	2.2126	2.1998	2.2885	2.2476	2.2789	2.1646
10	2.9664	2.231	2.0918	2.1727	2.2219	2.2215	2.1646
20	2.5182	2.1113	2.1180	2.3057	2.2763	2.2745	2.1646
50	2.2559	2.0106	2.1037	2.2813	2.3008	2.2571	2.1646
100	2.2783	2.2772	2.3185	2.2922	2.2950	2.3043	2.1646

Table 1: RMSE of LApf with respect to m and K over 100 Monte Carlo trials

	m = 6	10	20	100	200	500	1000
KNN = 3	0.0747	0.0589	0.0456	0.0102	0.0088	0.0096	0
4	0.0477	0.0408	0.0328	0.0109	0.0115	0.0113	0
5	0.0386	0.0329	0.0221	0.0127	0.0121	0.0118	0
6	0.0341	0.0245	0.0173	0.0129	0.0143	0.0137	0
7	0.0279	0.021	0.0146	0.0126	0.016	0.0137	0
8	0.0248	0.0183	0.0141	0.0137	0.0166	0.014	0
9	0.0242	0.0171	0.0139	0.0155	0.016	0.0144	0
10	0.0224	0.0159	0.0141	0.014	0.0165	0.0149	0
20	0.0186	0.0166	0.0150	0.0200	0.0191	0.0174	0
50	0.0193	0.0170	0.0199	0.0214	0.0213	0.0195	0
100	0.0209	0.0223	0.0252	0.0249	0.0230	0.0214	0

Table 2: Approximation error of LApf with respect to m and K over 100 Monte Carlo trials

m increases. On the other hand, at $K = 7$, there is a huge spike in runtime for all values of m. Fig. 6 shows the breakdown of total runtime for $m = 10$. It appears that the computational overhead of eigenvalue decomposition spikes at $K = 7$. It is not yet clear what causes this sudden spike. We omit the figures for other values of m to save space; but it is worth nothing that the eigenvalue decomposition accounts for the overwhelming majority of computational overhead in all cases.

	m = 6	10	20	100	200	500	1000
K = 3	46.621	45.8677	44.6806	47.9622	47.4402	47.213	47.9443
4	47.61	45.8321	46.9099	44.1239	46.8175	45.7268	48.0369
5	45.5334	45.9404	44.9823	44.2019	47.1917	46.8049	49.0212
6	47.1295	48.5234	47.4019	49.2646	50.1962	54.1456	58.088
7	59.0893	56.0343	58.6746	56.3333	58.1322	55.1389	49.3091
8	48.8316	47.9977	47.1873	45.6619	46.6635	47.3256	48.1953
9	49.4382	51.4236	48.8129	46.9926	47.2865	51.273	57.2445
10	46.4701	46.178	49.3883	50.8106	51.7062	49.7496	47.8554
20	45.1474	48.0359	42.6048	29.7810	31.1612	30.2740	30.4691
50	47.3068	53.4298	39.3070	31.5229	31.4748	31.1005	35.6691
100	55.7847	44.9831	39.8943	36.4267	32.4495	33.2130	34.6833

Table 3: Total runtime of LApf with respect to m and K over 100 Monte Carlo trials

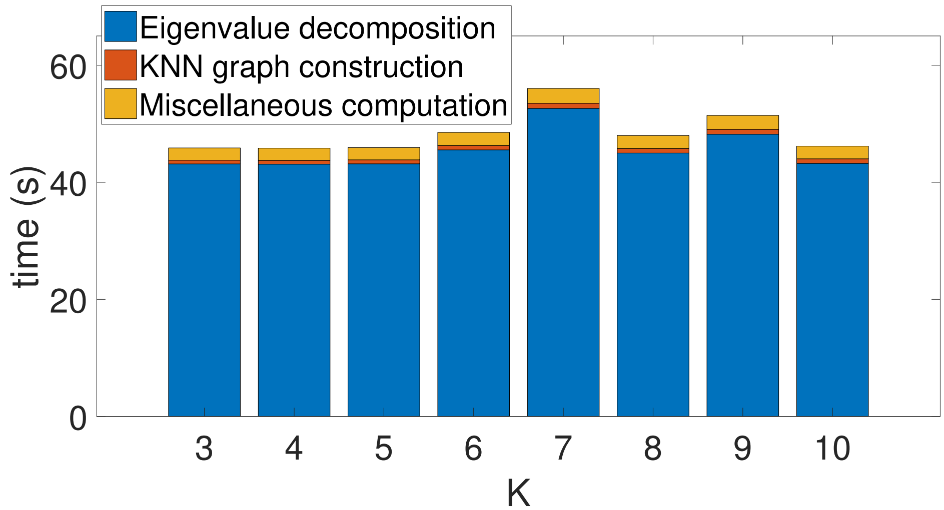


Figure 6: Breakdown of total runtime of LApf with respect to K for $m = 10$

In a distributed implementation of LApf, our objective is to minimize communication overhead without sacrificing tracking accuracy. Our simulation results suggest that this can be achieved by minimizing m and maximizing K.

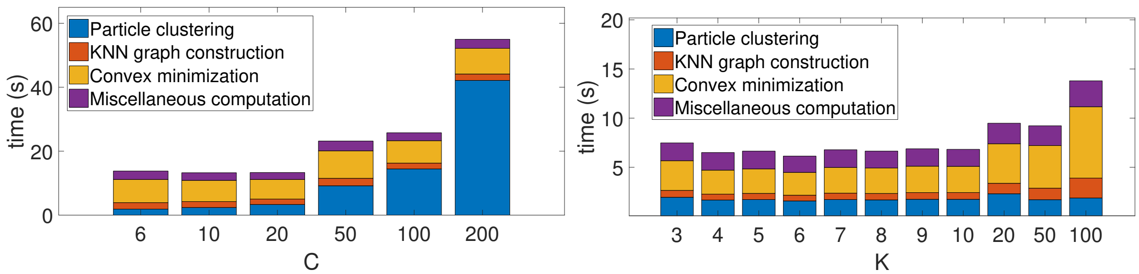
4.3.2 Cluster particle filter

The Clusterpf has two parameters of interest: number of clusters C and number of neighbors K for the graph construction. Table 4 shows the average RMSE of Clusterpf with respect to C and K. As C increases, the RMSE decreases. At $C = N = 1000$, the Clusterpf is equivalent to the centralized BSpf (assuming distributed summation without error) and the performance is independent of K. For any given value of C, as K increases, RMSE decreases. We note that these trends are similar to those shown in LApf. In fact, the parameter C serves the same role as m in LApf. Higher value of C requires higher communication overhead but reduces the approximation error of particle log-likelihoods. These results suggest that, as in the case of LApf, one may opt for a higher value of K in exchange for reduced communication overhead without significantly degrading tracking performance.

	C = 6	10	20	50	100	200	500	1000
K = 3	7.2659	5.8947	4.9396	4.2876	3.5259	2.9242	2.1867	2.145
4	5.788	4.8958	4.2774	3.8178	3.6676	2.7393	2.1351	2.145
5	4.8997	4.3572	4.2338	3.6465	3.3935	2.6175	2.082	2.145
6	4.484	3.5367	3.4824	3.3628	2.9538	2.3805	2.2021	2.145
7	3.8512	3.8754	3.7534	3.2243	2.6819	2.2898	2.1499	2.145
8	3.9089	3.3702	3.3426	2.9276	2.5027	2.2326	2.1574	2.145
9	3.5868	3.1086	3.0701	2.8636	2.5661	2.2189	2.161	2.145
10	3.51	3.0733	2.919	2.4383	2.3067	2.1811	2.1323	2.145
20	2.4083	2.2209	2.2114	2.2414	2.2362	2.2464	2.1979	2.145
50	2.2432	2.2315	2.1922	2.274	2.2721	2.2163	2.2494	2.145
100	2.3841	2.2779	2.2087	2.1232	2.1998	2.2225	2.233	2.145

Table 4: RMSE of Clusterpf with respect to K and C over 100 Monte Carlo trials

Table. 5 shows the total runtime of Clusterpf with respect to K and m. As C increases, so does the



(a) K=100 (b) C=6

Figure 7: Breakdown of runtime of Clusterpf over 100 Monte Carlo trials.

runtime with a huge spike after $C = 200$. A detailed breakdown of total runtime shows (Fig. 7(a)) that the computational overhead of particle clusters increases significantly with higher C. For a given value of C (Fig. 7(b)), increasing K leads to higher runtime which is caused by the higher computational overhead for nearest-neighbor graph construction and convex optimization.

	C = 6	10	20	50	100	200	500	1000
K = 3	7.5892	7.261	7.319	11.6321	19.1915	42.2104	194.2847	748.8655
4	5.3822	7.376	8.1377	10.3778	20.1636	41.0503	209.7392	745.8139
5	5.7585	7.7351	8.5754	11.9199	22.4751	41.4862	202.5959	737.2755
6	5.6528	6.1532	6.9711	12.2259	21.3191	47.8484	208.5728	731.9177
7	5.6642	6.2151	8.2362	13.5284	21.5673	45.5246	201.8637	735.6102
8	5.5074	7.2165	8.6544	13.8472	22.0559	46.9042	212.0588	752.7881
9	5.443	7.3545	8.7816	11.9092	20.5374	48.2595	205.2912	752.7088
10	5.47	7.4078	8.3934	13.7549	18.4591	46.0232	205.4603	742.8909
20	9.4839	8.0864	9.1319	13.1057	26.0044	42.5927	215.7453	773.2
50	9.2247	10.1492	11.2527	16.6921	24.2699	45.6228	205.8808	769.31
100	13.7989	13.2524	13.3149	23.1734	25.7608	54.9912	217.2478	766.9748

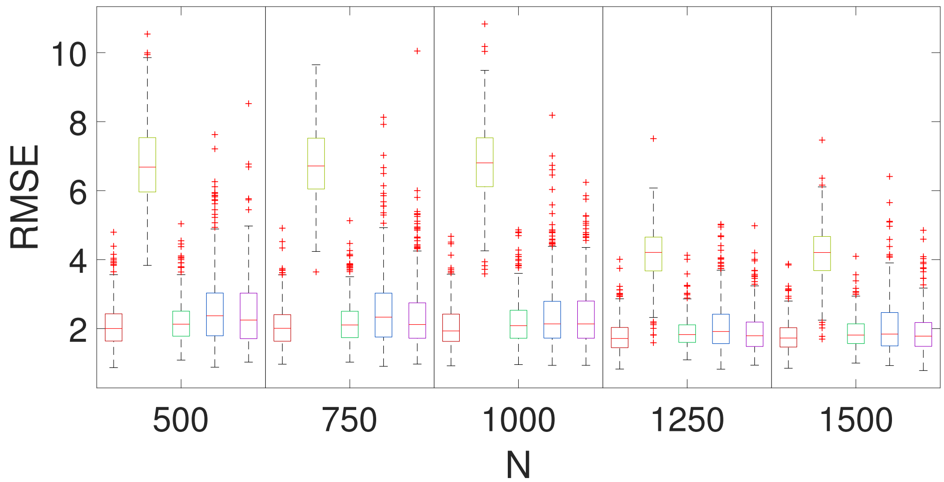
Table 5: Runtime of Clusterpf with respect to K and C over 100 Monte Carlo trials

4.4 Performance comparison between filters

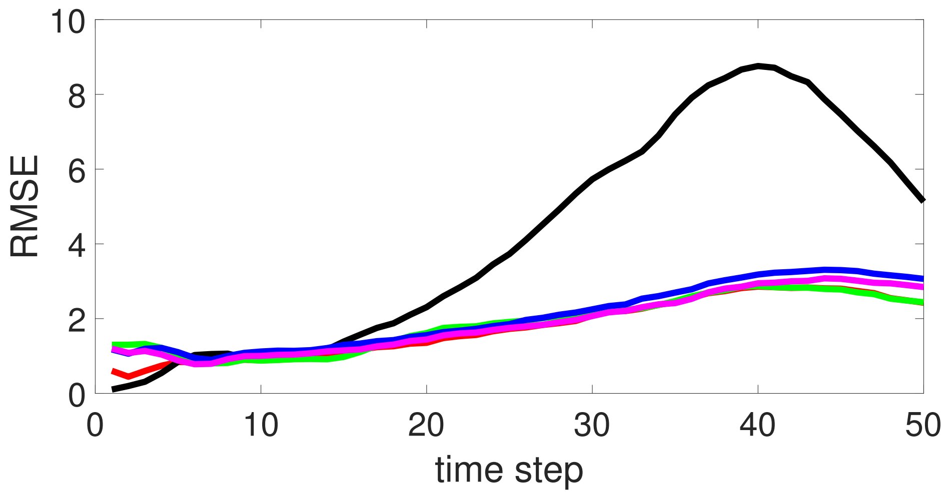
In this section, we compare the filters against each other. We use the same test track from the previous section. Based on our results from previous sections, we use the following algorithm-specific parameters: $d = 1$ (LCpf), $K = 100$, $m = 6$ (LApf) and $K = 100$, $C = 6$ (Clusterpf). These parameters are chosen to minimize the communication overhead without significantly degrading tracking performance. We note that CSSpf, LApf and Clusterpf have the same communication overhead (i.e., distributed aggregation of six statistics) whereas LCpf has a much higher overhead (i.e., aggregation of twenty scalars across sensors).

Fig. 8 shows the boxplot of RMSE with respect to N, the number of particles and the time-averaged RMSE for $N = 1000$. The CSSpf has the worst performance by a significant margin while the other four particle filters have similar performance and the performance of CSSpf starts degrading around time step $k = 15$. Fig. 9 shows the estimated trajectory of CSSpf from one trial and we can clearly see where the estimated track diverges from the true track. Table. 6 shows the average total runtime of all filters. BSpf, CSSpf and LCpf have similar runtime and are considerably faster than LApf and Clusterpf. The LApf has the longest runtime by far with the gap increasing at higher N which can be attributed to the eigenvalue decomposition.

In our next test, we shift the target trajectory and repeat the simulations. As Fig. 10 shows, all algorithms achieve lower RMSE compared to the first track. The CSSpf again has the highest RMSE; although the



(a) Boxplot of RMSE



(b) time-averaged RMSE for $N = 1000$

Figure 8: RMSE comparison of all particle filters. For each value of N, from left to right: BSpf, CSSpf, LCpf, LApf, Clusterpf

	N=500	N=750	N=1000	N=1250	N=1500
BSpf	0.2029	0.2425	0.2869	0.5624	0.5639
CSSpf	0.1265	0.1521	0.1825	0.6723	0.7253
LCpf	0.3053	0.3539	0.4112	0.7455	0.7796
LApf	7.6591	19.6260	39.5854	136.5413	247.4128
Clusterpf	5.0266	7.4632	10.4339	22.5435	29.3879

Table 6: Average total runtime of all filters with respect to N for 1st track

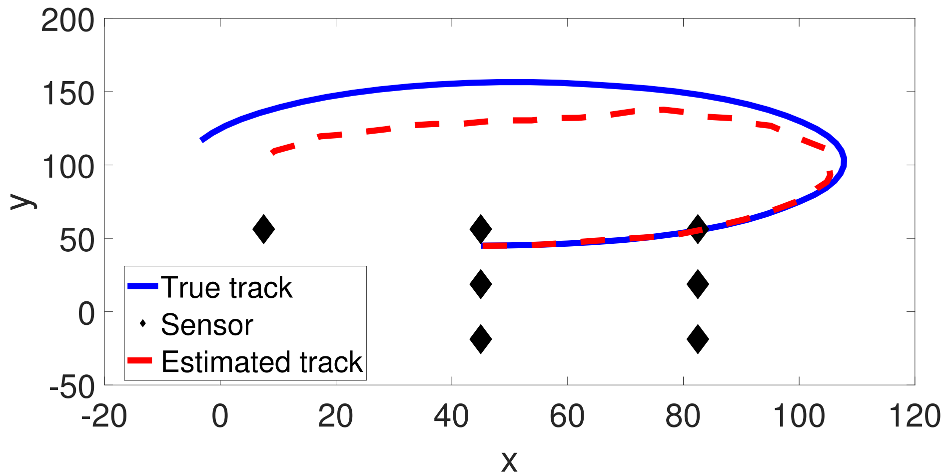


Figure 9: Estimated trajectory of CSSpf for one Monte Carlo trial at $N = 1000$

	N=500	N=750	N=1000	N=1250	N=1500
BSpf	0.2064	0.2495	0.3045	0.5407	0.5557
CSSpf	0.2649	0.3313	0.4012	0.6497	0.7179
LCpf	0.2947	0.3678	0.4372	0.7125	0.7657
LApf	7.7287	20.0299	41.4953	145.4905	265.9044
Clusterpf	5.0872	7.6359	10.9596	23.2325	31.8395

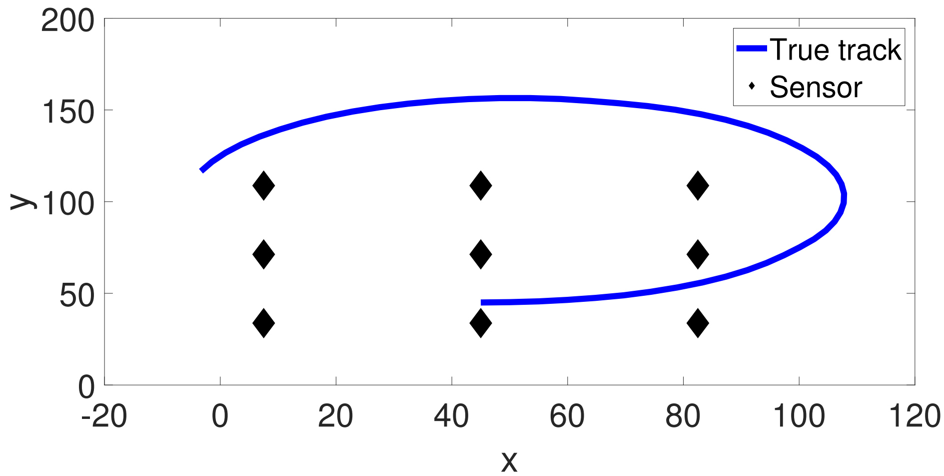
Table 7: Average total runtime of all filters with respect to N for 2nd track

performance gap is considerably smaller. For all values of N, all other filters have similar performance. Fig. 11 shows the estimated trajectory of CSSpf for one random trial and the divergence between estimated and true tracks again occur around the outer corner turn. Table 7 shows the average runtime. Again, BSpf, CSSpf and LCpf have similar short runtime while LApf has the highest runtime.

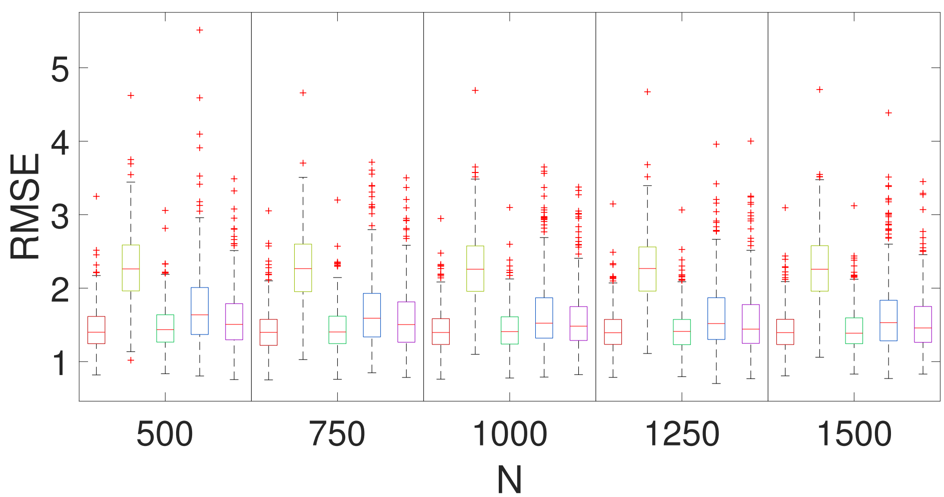
5 Conclusion

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) Track



(b) RMSE

Figure 10: 2nd Test trajectory and boxplots of RMSE with respect to N. For each value of N, from left to right: BSpf, CSSpf, LCpf, LApf, Clusterpf

References

[1] A. Mohammadi and A. Asif, “A constraint sufficient statistics based distributed particle filter for bearing only tracking,” in *IEEE Int. Conf. Communications (ICC)*, Ottawa, ON, Canada, Jun 2012, pp. 3670–3675.

[2] O. Hlinka, O. Sluciak, F. Hlawatsch, P. Djuric, and M. Rupp, “Likelihood consensus and its application to distributed particle filtering,” *IEEE Trans. Signal Process.*, vol. 60, no. 8, pp. 4334–4349, 2012.

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

[4] C. W. Chao, M. Rabbat, and S. Blouin, “Particle weight approximation with clustering for gossip-based distributed particle filters,” in *IEEE Int. Workshop Comp Comput. Advances Multi-Sensor Adaptive Process. (CAMSAP)*, Cancun, Mexico, Dec 2015, pp. 85–88.

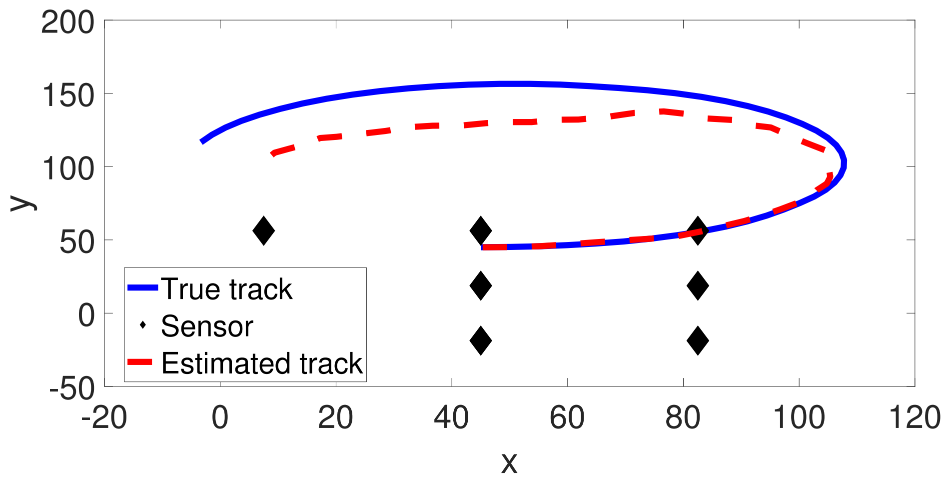


Figure 11: Estimated trajectory of CSSpf for one Monte Carlo trial at $N = 1000$