Distributed particle filter for bearing-only tracking

Jun Ye Yu

December 6, 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**, **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. 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 have 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:

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

where 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 $G_{s,j}$. We omit the detailed expressions for $G_{s,j}$ and $F_j(X)$, and refer interested readers to []].

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:

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

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. (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., nearby particles 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 the basis vectors of LApf are flexible and adapted to the particle locations whereas the LCpf requires fixed basis vectors. On the other hand, LApf requires synchronization of random number generator to ensure that all sensors have the same particle set.

3.4 Clustering particle filter

In Clusterpf, the particles are grouped into N_c clusters based on their position. The sensors just need to 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, 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.

As in the case of LApf, the Clusterpf requires that all sensors have the same particles.

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. The target starts near the top center sensor and travels 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 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 $1 - P_{cv} = 0.95$.

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

$$H_s(X(k)) = \arctan 2\left(\frac{x_t - x_s}{y_t - y_s}\right) + \eta(k) \tag{13}$$

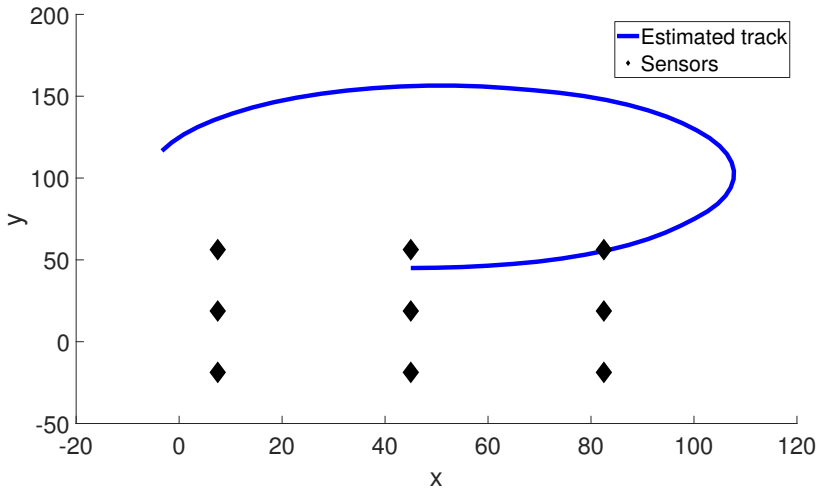


Figure 1: Target trajectory (blue curve) and sensor positions (red diamond)

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

$$Q = \sigma_a^2 \begin{bmatrix} \frac{T^3}{3} & 0\frac{T^2}{2} & 0 \\ 0 & \frac{T^3}{3} & 0\frac{T^2}{2} \\ \frac{T^2}{2} & 0T & 0 \\ 0 & \frac{T^2}{2} & 0T \end{bmatrix} \tag{14}$$

$$R = \sigma_\theta^2 \tag{15}$$

where $\sigma_a = 10^{-4}$, $T = 1$ and $\sigma_\theta = 0.0873$ rad $= 5$ degree.

4.2 Algorithm setup

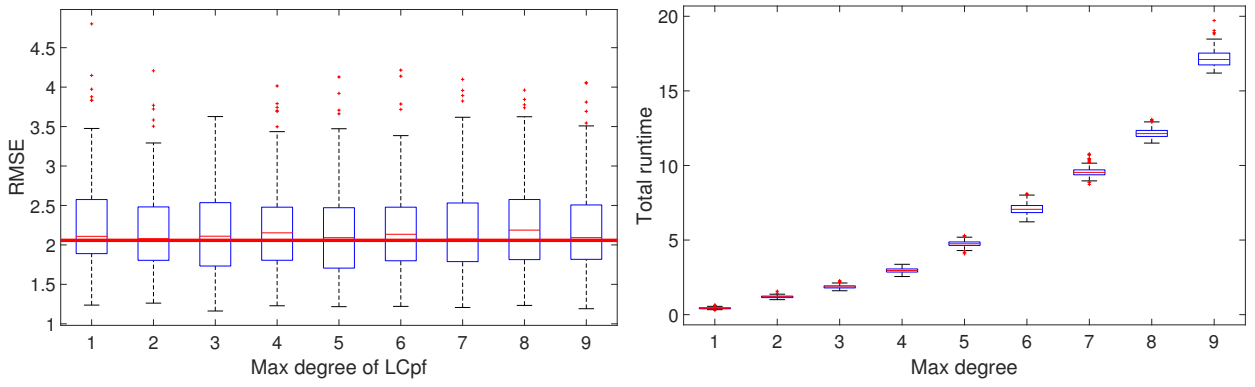
All particle filters use a total of $N = 1000$ particles. At time step 1, we generate the initial particles using the true target state: $X_i(1) \sim \mathcal{N}(X(1), R_{\text{initial}})$ with $R_{\text{initial}} = \text{diag}([0.5^2, 0.5^2, 0.05^2, 0.05^2])$.

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

$$\begin{aligned} \beta_1(X) &= x_t^0 y_t^0 = 1 \\ \beta_2(X) &= x_t^0 y_t^1 = y_t \\ \beta_3(X) &= x_t^0 y_t^2 = y_t^2 \\ \beta_4(X) &= x_t^1 y_t^0 = x_t \\ &\cdots \\ \beta_9(X) &= x_t^2 y_t^2 \end{aligned}$$

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.



(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.

4.3 Individual algorithm performance

In this section, we evaluate the performance of individual algorithms with respect to their specific parameters. We run a number of Monte Carlo simulations on each algorithm while varying one single parameter. 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). Our objective is to select a set of parameters that offer optimal trade-off between tracking performance and computational load.

4.3.1 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 leads to more basis functions and likely better approximation of the measurement model. On the other hand, more basis functions lead to more computation and longer runtime.

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 difference in RMSE for $1 \leq d \leq 9$. This suggests that the additional basis functions for $d > 1$ do not yield a better approximation of the measurement model $H(X_i)$.

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 $\sum_{i=1}^{N} \sum_{s=1}^{S} (H_s(X_i) - \hat{(H)}_s(X_i))/N$.

Fig. 4.3.1 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. Based on the computational overhead, we will select $d = 1$ for LCpf in subsequent simulations.

4.3.2 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

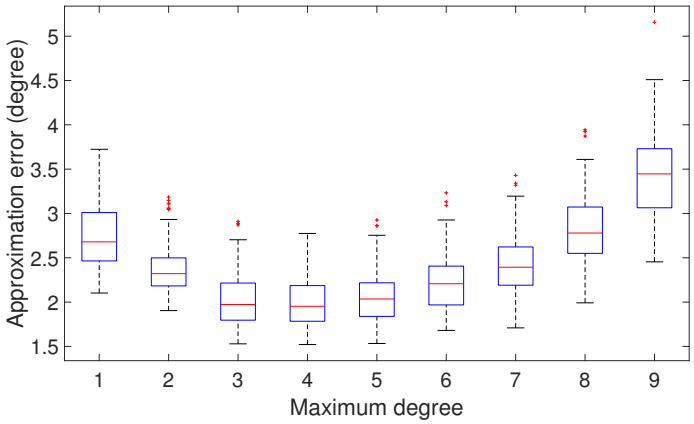


Figure 3: Boxplot of measurement approximation error with respect to max degree. All results are computed over 100 Monte Carlo trials

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 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 approximation error actually goes up with as K increases. It is worth noting that the high values of m ensure the approximation error remains small.

	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

Table. 3 shows the total runtime of LApf with respect to K and m. 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. On the other hand, at $K = 7$, there is a huge spike in runtime for all values of m. Fig. 4 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
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 = 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

4.4 Performance comparison between filters

To be completed

5 Conclusion

To be completed

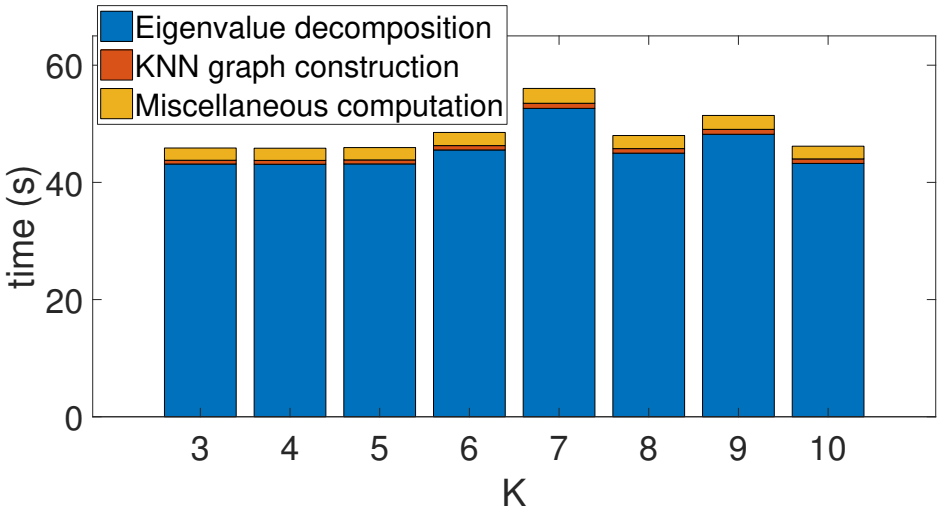


Figure 4: Breakdown of total runtime of LApf with respect to K for $m = 10$