Distributed likelihood consensus particle filter

Jun Ye Yu

November 8, 2017

1 Introduction

In this report we present a distributed single-target particle filter in which the joint log-likelihood function is approximated using *likelihood consensus* (LCPF). We compare the performance of the proposed filter against the centralized bootstrap particle filter (BPF) and another distributed particle filter based on constraint sufficient statistics (CSSPF).

2 Problem statement

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

3 Distributed particle filter

In Bayes' filter, we recursively estimate the posterior target density $f(X(k)|Z(1), ...Z(k))$ where $Z(k) = \{z^1(k)...z^S(k)\}$. In a particle filter, we model the target density using a set of N particles with normalized weights $\{X_i(k), w_i(k)\}_{i=1}^N$. Therefore, the objective is to recursively estimate the posterior particle weights. This in turn requires the computation of joint log-likelihood:

$$\log(f(z^1(k), ...z^S(k)|X_i(k))) = \sum_{s=1}^{S} \log\left(f(z^s(k)|X_i(k))\right) \tag{1}$$

where we assume that measurements from different sensors are conditionally independent given the target state.

A naive approach requires running N consensus algorithms in parallel for all N particle. This is clearly infeasible for large N. In this report, we approximate the log-likelihood function using likelihood consensus to devise a distributed filter with reduced communication overhead.

For the rest of the report, we omit the time index k when there is no ambiguity.

4 Likelihood consensus

We assume that the sensor measurement is corrupted by white Gaussian noise. This gives

$$\begin{aligned}\log(f(z^s|X_i) &\propto \frac{-(z^s - H^s(X_i))^2}{2\sigma^2} \\ &= \frac{-(z^s)^2 - H^s(X_i)^2 + 2z^s H^s(X_i)}{2\sigma^2}\end{aligned} \tag{2}$$

where σ is the standard deviation of the measurement noise.

The basic idea of likelihood consensus is to approximate the measurement model as follows:

$$H^s(X_i) \approx \sum_{j=1}^{J} \alpha_j^s \beta_j(X_i) \tag{3}$$

where $\beta_j(X_i)$ are basis functions that depend only on X_i and are known to all sensors, and the coefficients α_j^s encompass all the information of sensor s.

We plug Eq. (3) and Eq. (2) into Eq. (1) to obtain

$$\begin{aligned}\log(f(z^1(k), ...z^S(k)|X_i(k))) &\propto -\sum_{s=1}^{S} \frac{(z^s)^2}{2\sigma^2} - \sum_{s=1}^{S} \frac{\left(\sum_{j=1}^{J} \alpha_j^s \beta_j(X_i)\right)^2}{2\sigma^2} + \sum_{s=1}^{S} \frac{z^s \sum_{j=1}^{J} \alpha_j^s \beta_j(X_i)}{\sigma^2} \\ &= -\frac{\sum_{s=1}^{S}(z^s)^2}{2\sigma^2} - \sum_{j=1}^{J}\sum_{l=1}^{L} \frac{\sum_{s=1}^{S} \alpha_j^s \alpha_l^s \beta_j(X_i)\beta_l(X_i)}{2\sigma^2} + \sum_{j=1}^{J} \frac{\sum_{s=1}^{S} z^s \alpha_j^s \beta_j(X_i)}{\sigma^2} \\ &= -\frac{\sum_{s=1}^{S}(z^s)^2}{2\sigma^2} - \sum_{m=1}^{M} \psi_m^s(X_i) \frac{\sum_{s=1}^{S} \phi_m^s}{2\sigma^2} + \sum_{j=1}^{J} \beta_j(X_i) \frac{\sum_{s=1}^{S} z^s \alpha_j^s}{\sigma^2}\end{aligned} \tag{4}$$

where, for the last equality, we employ a suitable mapping $m \to (j, l)$, $\phi_m(X_i) = \beta_j(X_i)\beta_l(X_i)$ and $\psi_m^s = \alpha_j^s \alpha_l^s$.

Eq. (4) suggests that the joint log-likelihood can be constructed using $M + J$ consensus algorithms in parallel to compute the global sufficient statistics $\sum_{s=1}^{S} \psi_m^s$ and $\sum_{s=1}^{S} z^s \alpha_j^s$. The first term in Eq. (4) can be ignored since it is constant and equal for all particles.

4.1 Computation of coefficients

We construct the $N \times J$ matrix Φ as follows:

$$\begin{pmatrix} \beta_1(X_1) & ... & \beta_J(X_1) \\ ... & ... & ... \\ \beta_1(X_N) & ... & \beta_J(X_N) \end{pmatrix} \tag{5}$$

For each sensor s, we also construct the following $N \times 1$ row vector

$$\Lambda^s = [H^s(X_1), ...H^s(X_N)]^T \tag{6}$$

where T denotes the transpose operation.

Finally, the coefficients can be estimated as follows:

$$\alpha^s = [\alpha_1^s, ...\alpha_J^s]^T = (\Phi^T \Phi)^{-1} \Phi^T \Lambda^s \tag{7}$$

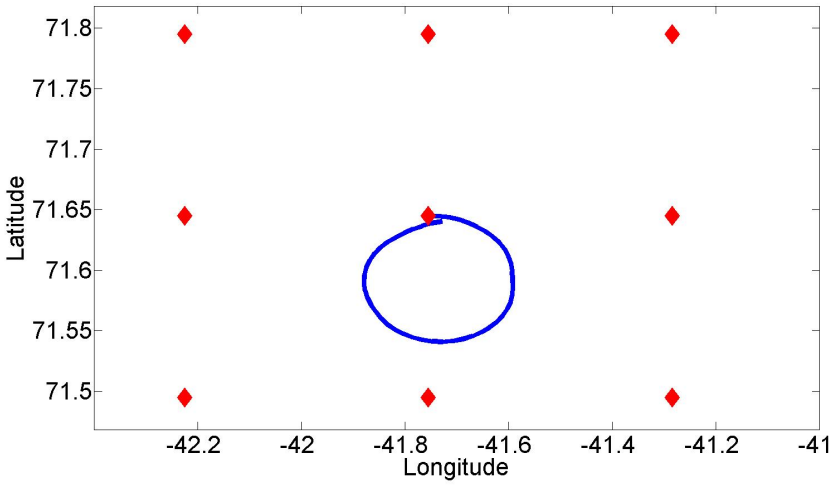

Figure 1: Target trajectory (blue curve) and sensor positions (red diamond)

5 Performance evaluation

5.1 Simulation setup

We compare the performance of likelihood consensus filter against that of a centralized bootstrap filter and a second distributed filter based on constraint sufficient statistics.

We construct a network of 9 sensors in a 3×3 grid. The target starts near the center sensor and travels in clockwise direction over 50 time steps. The sensors remain static over time. The target position is measured in longitude and latitude and the target velocity is measured in km/time step. Fig. 1 shows a sample target trajectory and sensor positions.

The simulated target randomly switches between two different motion models: constant velocity with probability P_{cv} and coordinated clockwise turn with probability $1 - P_{cv}$.

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

$$z^s(X) = \arctan 2 \left(\frac{\sin(x_t - x^s)\cos(y_t)}{\cos(y^s)\sin(y_t) - \sin(y^s)\cos(y_t)\cos(x_t - x^s)} \right) \tag{8}$$

We also inject additive white Gaussian motion and measurement noise with covariance Q and R respectively.

$$Q = \sigma_a^2 \begin{bmatrix} \frac{T^3}{3} & 0\frac{T^2}{2} & 0 \\ 0 & \frac{T^3}{3} & 0\frac{T^2}{2} \\ \frac{T^2}{2} & 0T & 0 \\ 0 & \frac{T^2}{2} & 0T \end{bmatrix} \tag{9}$$

$$R = \sigma_\theta^2 \tag{10}$$

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

5.2 Algorithm setup

All particle filters use a total of N particles. At time step 1, we generate the initial particles using the true target state: $X_i(1) \sim \mathcal{N}(X_i(1), X(1), R_{\text{initial}})$ with $R_{\text{initial}} = \text{diag}([10e^{-4}, 10e^{-4}, 10e^{-5}, 10e^{-5}])$.

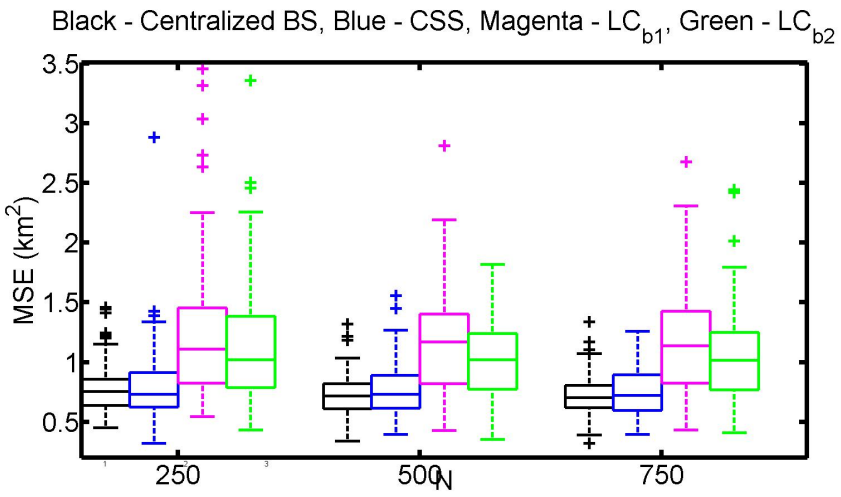

Figure 2: Mean squared position error with respect to number of particles

All three particle filters use the same dynamic model, and they only differ in the likelihood evaluation. We further assume that the random number generators are synchronized across the sensors.

For the CSSPF and LCPF, we compute the global sufficient statistics exactly (i.e., without gossiping or consensus) so that all difference in tracking performance comes from likelihood computation.

For LCPF, we consider two different sets of basis functions. Consider a set of points in the tracking area: $\mu_j = [x_j, y_j], j = 1...J$.

In the first set of basis function, we have

$$\beta_j(X) = \arctan 2 \left(\frac{\sin(x_t - x_j)\cos(y_t)}{\cos(y_j)\sin(y_t) - \sin(y_j)\cos(y_t)\cos(x_t - x_j)} \right) \tag{11}$$

In other words, we place a virtual sensor at each point μ_j and the basis function is simply the expected bearing measurement received at that virtual sensor.

In the second set of basis function, we have

$$\beta_j(X) = \mathcal{N}(x; \mu, R_\beta) \tag{12}$$

In other words, we treat μ_j as the mean of a multi-variate normal distribution.

We denote by LCBPF$_\mathbf{1}$ and LCBPF$_\mathbf{2}$ respectively the use of first and second set of basis functions.

5.3 Simulation performance

Fig. 2 shows the boxplots of mean squared position error with respect to the number of particles. All results are averaged over 100 Monte Carlo trials. At each trial, the trajectory and sensor measurements change. For any trial, all algorithms have the same initial particles and measurements.

The BSPF and CSSPF have similar tracking performance; however the LCPF has consistently the worst performance (i.e., almost double the MSE of the other methods).

6 Conclusion