Distributed Kalman Filtering for Sensor Networks

Author: Reza Olfati-Saber
Presented by: Ehsan Elhamifar,
Vision Lab, Johns Hopkins University

Distributed Kalman Filtering

- Distributed estimation and filtering is one of the most fundamental collaborative information processing problems in wireless sensor networks (WSN).

Distributed Kalman Filtering

- Distributed estimation and filtering is one of the most fundamental collaborative information processing problems in wireless sensor networks (WSN).
- We have a number of sensors observing a process which is not observable for each sensor, but the process is observable for the collection of sensors.

Distributed Kalman Filtering

- Distributed estimation and filtering is one of the most fundamental collaborative information processing problems in wireless sensor networks (WSN).
- We have a number of sensors observing a process which is not observable for each sensor, but the process is observable for the collection of sensors.
- Central Kalman Filter ($\hat{x}_c$) is computationally expensive!

Distributed Kalman Filtering

- Distributed estimation and filtering is one of the most fundamental collaborative information processing problems in wireless sensor networks (WSN).
- We have a number of sensors observing a process which is not observable for each sensor, but the process is observable for the collection of sensors.
- Central Kalman Filter ($\hat{x}_c$) is computationally expensive!
- Is it possible that each sensor estimate $\hat{x}_c$ based on only local information from its neighbors?

Distributed Kalman Filtering

- Distributed estimation and filtering is one of the most fundamental collaborative information processing problems in wireless sensor networks (WSN).
- We have a number of sensors observing a process which is not observable for each sensor, but the process is observable for the collection of sensors.
- Central Kalman Filter ($\hat{x}_c$) is computationally expensive!
- Is it possible that each sensor estimate $\hat{x}_c$ based on only local information from its neighbors? Yes!

Kalman Filtering for Sensor Networks

Consider a sensor network with n sensors, interconnected via a network $G = (V, E)$, undirected connected graph

Kalman Filtering for Sensor Networks

Consider a sensor network with n sensors, interconnected via a network $G = (V, E)$, undirected connected graph

- Process evolves according to:

$$x(k+1) = A_k x(k) + B_k w(k), x(0) \sim N(\bar{x}(0), P_0) \in \mathbb{R}^m$$

Kalman Filtering for Sensor Networks

Consider a sensor network with n sensors, interconnected via a network $G = (V, E)$, undirected connected graph

- Process evolves according to:

$$x(k+1) = A_k x(k) + B_k w(k), x(0) \sim N(\bar{x}(0), P_0) \in \mathbb{R}^m$$

- Sensing model for the ith sensor:

$$z_i(k) = H_i(k)x(k) + v_i(k), z_i \in \mathbb{R}^p$$

Kalman Filtering for Sensor Networks

Consider a sensor network with n sensors, interconnected via a network $G = (V, E)$, undirected connected graph

- Process evolves according to:

$$x(k+1) = A_k x(k) + B_k w(k), x(0) \sim N(\bar{x}(0), P_0) \in \mathbb{R}^m$$

- Sensing model for the ith sensor:

$$z_i(k) = H_i(k)x(k) + v_i(k), z_i \in \mathbb{R}^p$$

- $w(k), v_i(k)$ are zero mean white Gaussian noise (WGN) with

$$E[w(k)w(l)^\top] = Q(k)\delta_{kl}$$

$$E[v_i(k)v_j(l)^\top] = R_i(k)\delta_{kl}\delta_{ij}$$

Central Kalman Filter for Sensor Networks

- Let $z(k) = col(z_1(k), \cdots, z_n(k)) \in \mathbb{R}^{np}$ be the collective sensor data of the entire sensor network at time k.

Central Kalman Filter for Sensor Networks

- Let $z(k) = col(z_1(k), \cdots, z_n(k)) \in \mathbb{R}^{np}$ be the collective sensor data of the entire sensor network at time k.
- Given the information $Z_k = \{z(0), \cdots, z(k)\}$, we want to estimate the state of the process.

Central Kalman Filter for Sensor Networks

- Let $z(k) = col(z_1(k), \cdots, z_n(k)) \in \mathbb{R}^{np}$ be the collective sensor data of the entire sensor network at time k.
- Given the information $Z_k = \{z(0), \cdots, z(k)\}$, we want to estimate the state of the process.
- Define
  - estimate of the process state: $\hat{x}_k = E(x_k|Z_k), \bar{x}_k = E(x_k|Z_{k-1})$
  - estimate of the error covariance: $P_k = \Sigma_{k|k-1},\ M_k = \Sigma_{k|k}$

Central Kalman Filter for Sensor Networks

- Let $z(k) = col(z_1(k), \cdots, z_n(k)) \in \mathbb{R}^{np}$ be the collective sensor data of the entire sensor network at time k.
- Given the information $Z_k = \{z(0), \cdots, z(k)\}$, we want to estimate the state of the process.
- Define
  - estimate of the process state: $\hat{x}_k = E(x_k|Z_k), \bar{x}_k = E(x_k|Z_{k-1})$
  - estimate of the error covariance: $P_k = \Sigma_{k|k-1}$, $M_k = \Sigma_{k|k}$
- Thus we want to perform KF for the system:
  - $x(k+1) = A_k x(k) + B_k w(k)$
  - $z(k) = H_k x(k) + v_k$
  - with $H_k = col(H_1(k), \ldots, H_n(k))$, $v_k = col(v_1(k), \ldots, v_n(k))$, $R_k = diag(R_1(k), \ldots, R_n(k))$

Central Kalman Filter for Sensor Networks

Kalman Filter iterations for the sensor network would be of the form:

Central Kalman Filter for Sensor Networks

Kalman Filter iterations for the sensor network would be of the form:

- $M_k = (P_k^{-1} + H_k^\top R_k^{-1} H_k)^{-1}$
- $K_k = M_k H_k^\top R_k^{-1}$
- $\hat{x}(k) = \bar{x}(k) + K_k(z(k) - H_k \bar{x}(k))$
- $P(k+1) = A_k M_k A_k^\top + B_k Q_k B_k^\top$
- $\bar{x}(k+1) = A_k \hat{x}(k)$

Central Kalman Filter for Sensor Networks

Kalman Filter iterations for the sensor network would be of the form:

- $M_k = (P_k^{-1} + H_k^\top R_k^{-1} H_k)^{-1}$
- $K_k = M_k H_k^\top R_k^{-1}$
- $\hat{x}(k) = \bar{x}(k) + K_k(z(k) - H_k \bar{x}(k))$: central estimate $(\hat{x}_c)$
- $P(k+1) = A_k M_k A_k^\top + B_k Q_k B_k^\top$
- $\bar{x}(k+1) = A_k \hat{x}(k)$

Central Kalman Filter for Sensor Networks

Kalman Filter iterations for the sensor network would be of the form:

- $M_k = (P_k^{-1} + H_k^\top R_k^{-1} H_k)^{-1}$
- $K_k = M_k H_k^\top R_k^{-1}$
- $\hat{x}(k) = \bar{x}(k) + K_k(z(k) - H_k \bar{x}(k))$: central estimate $(\hat{x}_c)$
- $P(k+1) = A_k M_k A_k^\top + B_k Q_k B_k^\top$
- $\bar{x}(k+1) = A_k \hat{x}(k)$

Next: Perform distributed state estimation (or tracking) for the process

Distributed KF for Sensor Networks

Define two aggregate quantities:

Distributed KF for Sensor Networks

Define two aggregate quantities:

- Fused inverse-covariance matrices:
  $S(k) = 1/n \sum_i H_i^\top(k) R_i^{-1}(k) H_i(k)$

Distributed KF for Sensor Networks

Define two aggregate quantities:

- Fused inverse-covariance matrices: $S(k) = 1/n \sum_i H_i^\top(k) R_i^{-1}(k) H_i(k)$
- Fused sensor data: $y(k) = 1/n \sum_i H_i^\top(k) R_i^{-1}(k) z_i(k)$

Distributed KF for Sensor Networks

Define two aggregate quantities:

- Fused inverse-covariance matrices:
  $S(k) = 1/n \sum_i H_i^\top(k) R_i^{-1}(k) H_i(k)$
- Fused sensor data: $y(k) = 1/n \sum_i H_i^\top(k) R_i^{-1}(k) z_i(k)$

Transformed update equations for the Central KF:

- $M_\mu(k) = (P_\mu^{-1}(k) + S(k))^{-1}$
- $\hat{x}(k) = \bar{x}(k) + M_\mu(k)(y(k) - S(k)\bar{x}(k))$
- $P_\mu(k+1) = A_k M_\mu(k) A_k^\top + B_k Q_\mu(k) B_k^\top$
- $\bar{x}(k+1) = A_k \hat{x}(k)$
- where $M_\mu(k) = nM_k$, $Q_i(k) = nQ(k)$, $P_\mu(0) = nP_0$.

Distributed KF for Sensor Networks

Define two aggregate quantities:

- Fused inverse-covariance matrices: $S(k) = 1/n \sum_i H_i^\top(k) R_i^{-1}(k) H_i(k)$
- Fused sensor data: $y(k) = 1/n \sum_i H_i^\top(k) R_i^{-1}(k) z_i(k)$

Transformed update equations for the Central KF:

- $M_\mu(k) = (P_\mu^{-1}(k) + S(k))^{-1}$
- $\hat{x}(k) = \bar{x}(k) + M_\mu(k)(y(k) - S(k)\bar{x}(k))$
- $P_\mu(k+1) = A_k M_\mu(k) A_k^\top + B_k Q_\mu(k) B_k^\top$
- $\bar{x}(k+1) = A_k \hat{x}(k)$
- where $M_\mu(k) = nM_k$, $Q_i(k) = nQ(k)$, $P_\mu(0) = nP_0$.

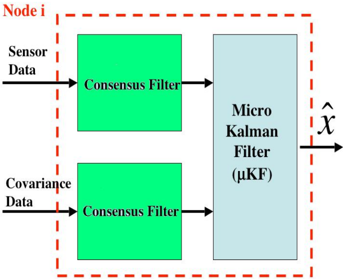
If each sensor implements a KF with above iterations, then all nodes have the same estimates as central estimate. Is it a DKF?

DKF for Sensor Networks

- If each node can compute the averages $y(k)$ and $S(k)$, a distributed KF emerges!

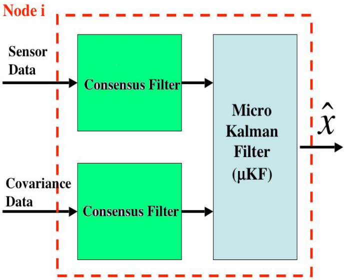
DKF for Sensor Networks

- If each node can compute the averages $y(k)$ and $S(k)$, a distributed KF emerges!



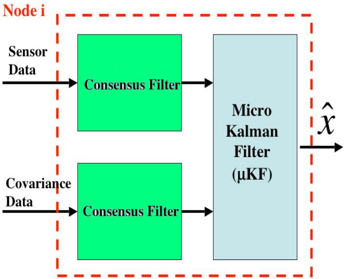
DKF for Sensor Networks

- If each node can compute the averages $y(k)$ and $S(k)$, a distributed KF emerges!
- Two consensus filters to compute $S(k)$ and $y(k)$ at each node using local information.



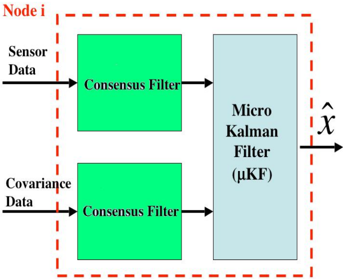
DKF for Sensor Networks

- If each node can compute the averages $y(k)$ and $S(k)$, a distributed KF emerges!
- Two consensus filters to compute $S(k)$ and $y(k)$ at each node using local information.
- Each node of the distributed Kalman filter that provides a state estimate is called a Micro-Filter.



DKF for Sensor Networks

- If each node can compute the averages $y(k)$ and $S(k)$, a distributed KF emerges!
- Two consensus filters to compute $S(k)$ and $y(k)$ at each node using local information.
- Each node of the distributed Kalman filter that provides a state estimate is called a Micro-Filter.
- Microfilters have identical structures.



Consensus Filters for DKF

- Use highpass consensus filters of the form
  - $\dot{q}_i = \beta \sum_{j \in N_i} (q_j - q_i) + \beta \sum_{j \in N_i} (u_j - u_i), \;\; \beta > 0$
  - $p_i = q_i + u_i$
  - where $\beta \sim O(1/\lambda_2)$ is relatively large.

Consensus Filters for DKF

- Use highpass consensus filters of the form
  - $\dot{q}_i = \beta \sum_{j \in N_i} (q_j - q_i) + \beta \sum_{j \in N_i} (u_j - u_i), \ \ \beta > 0$
  - $p_i = q_i + u_i$
  - where $\beta \sim O(1/\lambda_2)$ is relatively large.
- u denotes the input for each node:
  - For CF1 [$\rightarrow y(k)$]: $u_j = H_j^\top R_j^{-1} z_j$
  - For CF2 [$\rightarrow S(k)$]: $u_j = H_j^\top R_j^{-1} H_j$

Consensus Filters for DKF

- Use highpass consensus filters of the form
  - $\dot{q}_i = \beta \sum_{j \in N_i} (q_j - q_i) + \beta \sum_{j \in N_i} (u_j - u_i), \;\; \beta > 0$
  - $p_i = q_i + u_i$
  - where $\beta \sim O(1/\lambda_2)$ is relatively large.
- u denotes the input for each node:
  - For CF1 [$\rightarrow y(k)$]: $u_j = H_j^\top R_j^{-1} z_j$
  - For CF2 [$\rightarrow S(k)$]: $u_j = H_j^\top R_j^{-1} H_j$
- It is shown that for a connected network, outputs $p_i^1(k)$ and $p_i^2(k)$ of the highpass consensus filters asymptotically converge to $y(k)$ and $S(k)$.

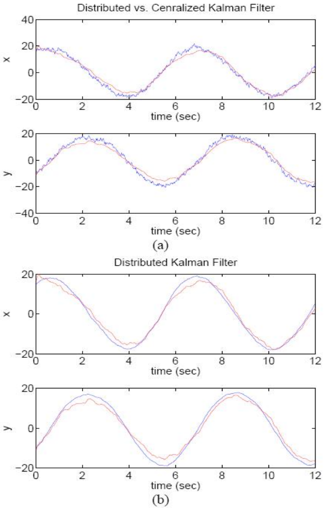
DKF for Sensor Networks

- Node i sends the message: $msg_i = (q_i^1(k), q_i^2(k), u_i^1(k), u_i^2(k))$ to all of its neighbors. $\Rightarrow$ Message size is of dimension $O(m(m+1))$ with m being the dimension of the state x.

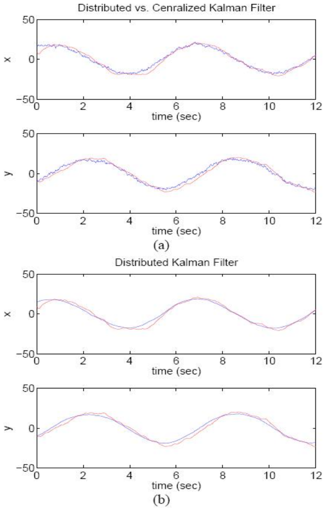
DKF for Sensor Networks

- Node i sends the message: $msg_i = (q_i^1(k), q_i^2(k), u_i^1(k), u_i^2(k))$ to all of its neighbors. $\Rightarrow$ Message size is of dimension $O(m(m+1))$ with m being the dimension of the state x.
- The communication scheme is fully compatible with packet-based communication in real-world WSN.

DKF Results



Distributed position estimation for a moving object by node $i = 100$: (a) DKF vs. KF (DKF is the smooth curve in red) and (b) Distributed Kalman filter estimate (in red) vs. the actual position of the object (in blue).



Distributed position estimation for a moving object by node $i = 25$: (a) DKF vs. KF (DKF is the smooth curve in red) and (b) Distributed Kalman filter estimate (in red) vs. the actual position of the object (in blue).