Distributed estimation in sensor networks

A. Benavoli

Dpt. di Sistemi e Informatica
Università di Firenze, Italy.
e-mail: benavoli@dsi.unifi.it
1. Sensor Networks
   - An introduction to Sensor Networks
   - Network architectures
   - Distributed estimation

2. Communication strategies

3. Follow on
In recent years, great attention has been devoted to multisensor data fusion for both military and civilian applications. *Data fusion techniques combine data from multiple sensors and related information to achieve more specific inferences than could be achieved by using a single, independent sensor.*

Civilian applications:
- monitoring of manufacturing processes;
- robotics;
- medical applications / environmental monitoring.

Military applications:
- target recognition;
- guidance for autonomous vehicles;
- battlefield surveillance;

A. Benavoli

Fully Decentralized Networks
An introduction to Sensor Networks

What Is Data Fusion?

“Data fusion is the process of combining data or information to estimate or predict entity states” [1].

What type of information is fused?

There are many types of sensors.

Low cost sensors:
- microphone/photo cell;
- thermostat;
- accelerometer;
- chemical;

Sophisticated sensors:
- radar;
- satellite.
An introduction to Sensor Networks

Sensor networks present many advantages over a single sensor:

- are dislocated over large regions;
- provide diverse characteristics/viewing angles of the observed phenomenon;
- are more robust to failures;
- gather many observations $\Rightarrow$ more data that, once fused, provide a more complete picture about the observed phenomenon.

There are many architectures to process the sensor data [2]:

- centralized;
- hierarchical (with feedback or without feedback);
- distributed.
Network architectures

- **Centralized**
- **Hierarchical Without Feedback**
- **Hierarchical With Feedback**
- **Distributed**

- **S** Sensor / data source
- **C** Information consumer
- **F** Fusion node

A. Benavoli
Fully Decentralized Networks
Centralized architecture

Data from multiple sensors are sent to a central fusion node where the data are fused.

Pros:
- fusion rule is easy;
- fusion rule is optimal;
- sensors are cheaper ⇒ don’t have computational capability.

Cons:
- it requires a large communication bandwidth;
- sent data include also clutter;
- the central node is a single point of failure.
Hierarchical architecture:
The fusion nodes are arranged in a hierarchy with the lowest level nodes **processing** sensor data and sending results to higher level nodes to be fused.
This architecture is with feedback if the higher-level nodes send the fused data back to the lower-level nodes.

Distributed architecture:
In a distributed architecture, there is not fixed master/slave relationship. Each node can communicate with other nodes subject to connectivity constraints. In the extreme case (fully distributed network), each sensor has its own processor to fuse the local data and cooperate with other sensor nodes (p2p network).
Hierarchical/Decentralized architectures 2/3

Pros:

- lighter processing load at each fusion node due to the distribution over multiple nodes;
- lower communication load since data does not have to be sent to a central processing node;
- higher survivability since there is no single point of failure associated with a central fusion node;

Cons → Technical Issues
Hierarchical/Decentralized architectures 3/3

Technical Issues:

- **Architecture**: how the nodes should share the fusion responsibility, e.g., which sources or sensors should report to each node, and the targets that each node should be responsible for.

- **Communication**: how the nodes should communicate, e.g., connectivity and bandwidth of the communication network, information push or pull, and communicating raw data versus processing results.

- **Fusion algorithms**: how the nodes should fuse data for high performance results and select their communication actions (who, when, what, and how).
The data collected from the sensors can be used:

- for making a decision on a hypothesis (detecting the presence of targets or classifying a signal);
- for estimation (target tracking).

This presentation will focus on distributed estimation and its application to target tracking problems.
An introduction to distributed estimation

In the target tracking the key problem is estimating the target state given the associated measurements. This target state may involve continuous variables such as position and velocity and discrete variables such as target type.

Consider a linear Gaussian model in a network of $N$ sensors:

$$
x_{k+1} = Fx_k + w_k
$$  \hspace{1cm} (1)

$$
y_{k}^{m} = H^{m}x_k + v_{k}^{m} \quad m = 1, 2, \ldots, N
$$  \hspace{1cm} (2)

with $E[w_k] = 0$, $E[w_kw_{l}'] = Q\delta_{kl}$, $E[v_{k}^{m}] = 0$, $E[v_{k}^{m}v_{l}^{m'}] = R^{m}\delta_{kl}$, $E[v_{k}^{i}v_{k}^{i'}] = 0$.

If all the sensor data are collected from the fusion node (centralized architecture), the Kalman filter gives the optimal solution for the distributed estimation problem.
To take into account the $N$ observations from the sensors:

$$\text{global observation } \Rightarrow \ y_k = Hx_k + v_k$$  \hspace{1cm} (3)

where $y_k = [y_k^1, y_k^2, \ldots, y_k^N]'$, $H = [H^1', H^2', \ldots, H^N']'$, $v_k = [v_k^1', v_k^2', \ldots, v_k^N']$ and $R = \text{diag}[R^1, R^2, \ldots, R^N]$.

The global estimate with all the sensor data is given by the KF:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + P_{k|k} H' R^{-1} [y_k - H\hat{x}_{k|k-1}]$$ \hspace{1cm} (4)

$$P^{-1}_{k|k} = P^{-1}_{k|k-1} + H' R^{-1} H$$ \hspace{1cm} (5)

$$\hat{x}_{k+1|k} = F\hat{x}_{k|k}$$ \hspace{1cm} (6)

$$\hat{P}_{k+1|k} = FP_{k|k} F' + Q$$ \hspace{1cm} (7)

with $\hat{x}_{k|k} = E[x_k|y_0, \ldots, y_k]$, $P_{k|k} = E[(x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k})'|y_0, \ldots, y_k]$. 
Kalman filter in Information form 2/3

Notice that:

\[
H' R^{-1} H = \sum_{i=1}^{N} H_i' R_i^{-1} H_i = \sum_{i=1}^{N} \left[ P_{i \mid k}^{-1} - P_{i \mid k-1}^{-1} \right] \tag{8}
\]

Hence it follows that

\[
P_{k \mid k}^{-1} = P_{k \mid k-1}^{-1} + \sum_{i=1}^{N} \left[ P_{i \mid k}^{-1} - P_{i \mid k-1}^{-1} \right] \text{ ill-conditioned} \tag{9}
\]

\[
P_{k \mid k}^{-1} \hat{x}_{k \mid k} = P_{k \mid k-1}^{-1} \hat{x}_{k \mid k-1} + \sum_{i=1}^{N} \left[ P_{i \mid k}^{-1} \hat{x}_{i \mid k} - P_{i \mid k-1}^{-1} \hat{x}_{i \mid k-1} \right] \tag{10}
\]
Therefore if the nodes in the network:

1. compute local estimates $\hat{x}_{k|k}^i$ and $P_{k|k}^i$;
2. transmit these estimates to the fusion node;

and if the fusion node computes the global estimate according to (9) and (10) then the fused estimate is equivalent to the estimate which can be obtained transmitting the measurements instead of the estimates. Therefore the fused estimate is optimal.

In a distributed architecture with

- broadcast communication;
- full communication rate;
- synchronized nodes (measurement $\rightarrow$ local estimate $\rightarrow$ communication $\rightarrow$ fusion);

equations (9) and (10) hold and the resulting estimates are therefore optimal.
Generic networks

Analyzing the fusion equation (10)

\[
P_{k|k-1}^{-1} \hat{x}_{k|k} = P_{k|k-1}^{-1} \hat{x}_{k|k-1} \sum_{i=1}^{N} \left[ P_{k|k-1}^{-1} \hat{x}_{k|k-1} - P_{k|k-1}^{-1} \hat{x}_{k|k-1} \right] \tag{11}
\]

\[
= P_{k|k-1}^{-1} \hat{x}_{k|k-1} + \sum_{i=1}^{N} H_i^i R_i^{-1} y_k \tag{12}
\]

the main problem is to remove the common information in order to avoid double-counting. For generic networks and state models it is hard to locate the common information.

There are two problems:

- common prior information ⇒ nodes share common prior information;
- common process noise ⇒ estimation errors from different sensors are not independent.
Common Prior Information

Nodes share common measurements.

\[ \hat{x}(y_1) \quad \hat{x}(y_1, y_2) \quad \hat{x}(y_1, y_3) \]
Consider a network of $m = 2$ sensors and the previous state/measurement model [3]:

$$x_{k+1} = Fx_k + w_k$$  \hspace{1cm} (13)

$$y^m_k = H^m x_k + v^m_k \quad m = i, j$$  \hspace{1cm} (14)

The state estimation using the measurement from sensor $m$ is

$$\hat{x}^m_{k|k} = F\hat{x}^m_{k-1|k-1} + K^m_k[y^m_k - H^m F\hat{x}^m_{k-1|k-1}]$$  \hspace{1cm} (15)

and the corresponding estimation error is

$$\tilde{x}^m_{k|k} = x_{k|k} - \hat{x}^m_{k|k} = [I - K^m_k H]F\tilde{x}^m_{k-1|k-1} + [I - K^m_k H]w_{k-1} - K^m_k v^m_k$$  \hspace{1cm} (16)

Hence, the cross-covariance is

$$P^{ij}_{k|k} = E[\tilde{x}^i_{k|k} \tilde{x}^j_{k|k}] = [I - K^i_k H][F\tilde{P}^{ij}_{k-1|k-1} F' + Q][I - K^j_k H]'$$  \hspace{1cm} (17)
Convex Combination (CC) and Covariance Intersection (CI):
- Require state and covariance matrix to be communicated.
- Consider neither common process noise nor common prior.

Bar-Shalom/Campo state vector combination (BC):
- Requires state, covariance matrix and gains to be communicated.
- Considers common process noise but not common prior.

Information fusion (IF):
- Requires state, cov. matrix and information graph to be communicated.
- Considers common prior but not common process noise.
- Optimal in a broadcast net with full-rate communication or in a generic net if the state is static or if the process noise is zero.
Given two state estimates and error covariance matrices \( \hat{x}_k^1 \), \( \hat{x}_k^2 \) and \( P_k^1 \), \( P_k^2 \):

\[
\hat{x}_{\text{fus}} = \bar{x} + P_{xz} P_{zz}^{-1} (z - \bar{z}) \quad \text{and} \quad P_{\text{fus}} = P_{xx} - P_{xz} P_{zz}^{-1} P_{zx}
\]

1. **CC and CI:**

\[
P_{\text{fus}} = \left( \sum_{i=1}^{2} \omega_i \left( P_i \right)^{-1} \right)^{-1} 
\]

\[
\hat{x}_{\text{fus}} = P_{\text{fus}} \left( \sum_{i=1}^{2} \omega_i \left( P_i \right)^{-1} \hat{x}_i \right)
\]

where \( \omega_i \) is a number between 0 and 1 such that:

- CC \( \Rightarrow \omega_i = 1 \) for each \( i \) or CI \( \sum_{i=1}^{2} \omega_i = 1 \).

2. **BC [3]:**

\[
P_{\text{fus}} = P_i - [P_i - P^{ij}][P_i + P_j - P^{ij} - P^{ji}][P_i - P^{ij}]^{-1}[P_i - P^{ji}]
\]

\[
\hat{x}_{\text{fus}} = \hat{x}_i + [P_i - P^{ij}][P_i + P_j - P^{ij} - P^{ji}][P_i - P^{ji}]^{-1}[\hat{x}_j - \hat{x}_i]
\]
Information fusion

Information fusion is developed from information filter. Consider a network with 2 nodes and the measurement vectors \( Y_1 = \{ y_1^1, \ldots, y_k^1 \} \) for node 1 and \( Y_2 = \{ y_1^2, \ldots, y_k^2 \} \) for node 2. The fusion equation is given by

\[
p(x|Y_1 \cup Y_2) = \frac{1}{c} \frac{p(x|Y_1)p(x|Y_2)}{p(x|Y_1 \cap Y_2)}
\]  
(22)

Linear Gaussian case:

\[
P_{\text{fus}} = \left( (P^1)^{-1} + (P^2)^{-1} - (\bar{P})^{-1} \right)^{-1} \]

\[
\hat{x}_{\text{fus}} = P_{\text{fus}} \left( P_1^{1^{-1}} \hat{x}^1 + P_2^{2^{-1}} \hat{x}^2 - \bar{P}^{-1} \bar{x} \right)
\]

(23)  
(24)

Notice that: these equations are not optimal for a generic network due to common process noise.
Observations

IF is optimal in a broadcast network with full communication rate. In wireless networks the communication is not broadcast:

- broadcast communication is more expensive in terms of energy;
- the communication range is limited ⇒ a node cannot transmit its estimate to all nodes in the networks.

The communication rate is not full. Nodes transmit their estimates occasionally

- to save communication bandwidth;
- to save energy.

If we use IF in this case, the fused estimate could diverge. For this type of networks the search of an optimal estimates fusion equation is still an open problem. CC and CI are used in practice to perform the fusion.
Consider this target model

\[
\begin{bmatrix}
    x^+ \\
    v_x^+ \\
    y^+ \\
    v_y^+
\end{bmatrix}
= \begin{bmatrix}
    1 & T & 0 & 0 \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 1 & T \\
    0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    x \\
    v_x \\
    y \\
    v_y
\end{bmatrix}
+ w
\]

\[
z^i = \begin{bmatrix}
    x \\
    y
\end{bmatrix}
+ v^i
\]

where:

- \( T \) is the scan period;
- \( w = w(k) \) is the process noise, zero-mean and with covariance \( Q \);
- \( v \) is the measurement noise, zero-mean and with covariance \( R = \text{diag}\{(0.05\text{km})^2, (0.05\text{km})^2\} \).
Simulation scenario:
- 1 maneuvering target;
- $N = 6$ nodes;
- broadcast communication;
- $P_d = 1$ and $P_{fa} = 0$. 
Comparison of fusion algorithms 3/3

Simulation results (300 Monte Carlo runs):

<table>
<thead>
<tr>
<th></th>
<th>$N = 6$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SS</td>
</tr>
<tr>
<td>$x (km)^2$</td>
<td>0.05930</td>
</tr>
<tr>
<td>$y (km)^2$</td>
<td>0.05960</td>
</tr>
<tr>
<td>$v_x (km/s)^2$</td>
<td>0.00130</td>
</tr>
<tr>
<td>$v_y (km/s)^2$</td>
<td>0.00150</td>
</tr>
</tbody>
</table>

Table: Simulation results - broadcast network - MSE

Similar results can be obtained changing the number of nodes ($N = 4, N = 8$) and the target trajectory.
Select a fusion algorithm

Considering that [6]:

- BC and IF are not optimal for generic networks;
- BC and IF are more expensive in terms of communication/computational cost;
- IF can diverge (if the network is not a broadcast net with full-rate communication);
- CC, BC and IF have similar performance.

⇓

CC/CI are a good compromise between performance, practicality and cost.
Communication strategies

Every node uses energy and communication bandwidth to communicate with its neighbors.

⇓

Reducing the communication rate is a way to save energy and bandwidth.

⇓

Issue: lower communication rate ⇒ lower tracking performance.

⇓

A good communication strategy is a strategy that takes into account both communication rate and tracking performance.
Metrics to evaluate the tracking performance

What is a good communication strategy?

strategy $\Rightarrow \left\{ \begin{array}{c} \min f(MSE_1, MSE_2, \ldots, MSE_{N_a}) \Rightarrow \text{tracking perfor.} \\ \min g(C_1, C_2, \ldots, C_{N_a}) \Rightarrow \text{communication perfor.} \end{array} \right.$

where $N_a$ is the number of active nodes, $MSE_i$ and $C_i$ are the mean square error and the communication cost for the i-th node.

\[
f(MSE_1, MSE_2, \ldots, MSE_{N_a}) = \begin{cases} 
\frac{1}{N_a} \sum_{i=1}^{N_a} MSE_i \\
\max_{i=1,\ldots,N_a} MSE_i
\end{cases}
\]

\[
g(C_1, C_2, \ldots, C_{N_a}) = \begin{cases} 
\frac{1}{N_a} \sum_{i=1}^{N_a} C_i \\
\max_{i=1,\ldots,N_a} C_i
\end{cases}
\]

A good strategy $\Rightarrow \min h(f, g)$ TRADEOFF.
Communication strategies

1. random communication;
2. transmit if good;
3. receive if bad;

**Strategy 1**

\[ \text{if } \text{rand}(1) < \text{thresh. then transmit} \]

**Strategy 2**

\[ \text{if } \text{error} < \text{thresh. then transmit} \]
\[ \text{where } \text{error} = k \text{ error} + (1-k)(z-\hat{z})' R^{-1}(z-\hat{z}) \]

**Strategy 3**

\[ \text{if } \text{error} > \text{thresh. then request} \]
\[ \text{where } \text{error} = k \text{ error} + (1-k)(z-\hat{z})' R^{-1}(z-\hat{z}) \]
Performance comparison 1/2

Simulation scenario:
- 1 target (maneuvering/not maneuvering);
- $P_d = 1$ and $P_{fa} = 0$;
- various network architectures (different number of nodes/links per node);

Simulation results:
fixed the communication cost (i.e. $g$ is the same for every strategy)

\[ \text{MSE strategy 1} \leq 15\text{-}20\% < \text{MSE strategy 2} \leq \text{MSE strategy 3} \]

Why?
Strategy 1 is better since track fusion is made more frequently.
strategy 1 $\Rightarrow$ $\max \bar{\Delta}_k = \frac{1}{N_a} \sum_{i=1}^{N_a} \Delta^i_k$

where $\Delta^i_k$ is 1 if the i-th node has received estimates from its neighbors at the scan $k$ or 0 otherwise.
Notice that $\Delta$ is different from $g(C_1, C_2, \ldots, C_{N_a}) = \frac{1}{N_a} \sum_{i=1}^{N_a} C_i$.

To define a good strategy we must also take into account $\Delta$.

**Collaborative strategy.**
- transmit if the $\Delta$ of my neighbor is low;
- else transmit at random.
Find an optimal fusion algorithm for generic network architectures or find performance bounds for the existing fusion rules (CRLB).

Derive new communication strategies considering also bandwidth constraints.

Given a limited bandwidth budget:

1. fixed bandwidth constraints on the communication links, or
2. time-varying bandwidth constraints on the communication links;

how should we choose the communication strategy (i.e. when/what transmit) and the fusion rule optimally so as to maximize the global performance?
References


