Intelligent distributed systems

Daniele Fontanelli

Department of Industrial Engineering
University of Trento
E-mail address: *daniele.fontanelli@unitn.it*

2022/2023

Outline

1. An example of distributed estimation
   - A simplified example
   - A simplified example with relative measurements

Outline

Outline

1. An example of distributed estimation
   - A simplified example
   - A simplified example with relative measurements

Toy example

Consider the following toy example:

- There are two points x_1 and x_2 moving on a line, starting from unknown initial conditions $x_1(0)$ and $x_2(0)$ and subject to a first order discrete dynamic affected by noise:

$$x_i(k+1) = x_i(k) + a_i d_i(k) + b_i \varepsilon_i(k), \quad k = 0, 1, 2, \dots,$$

  where $d_i(k)$ is the linear (known) displacement and $\varepsilon_i(k) \sim \mathcal{N}(0, \sigma_i^2)$ is the uncertainty affecting the displacement, supposed to be white.

Toy example

- Suppose that every $n_i \geq 1$ time steps, each point receives its *absolute* position measurement from an *exogenous* sensor, e.g. GPS, landmark detection (e.g. visual feature on a map or RFID position), anchor-based position, etc., with a certain uncertainty

$$z_i(n_i k) = x_i(n_i k) + \eta_i(n_i k), \quad k = 0, 1, 2, \ldots,$$

where $\eta_i(k) \sim \mathcal{N}(0, \xi_i^2)$ is a white Gaussian process.

Toy example

To properly estimate the point positions, it is possible to carry out an *(intermittent) Kalman filter* (provided that the two noise sequences are *uncorrelated*).
Intermittency takes place whenever $n_i > 1$, i.e. an *update step* every n_i *prediction steps*.
In other words, the estimator acts *alternatively* as a *predictor* (when there is no *update step*) or as a *filter*.

Toy example

Kalman filter

Recall that:

- *Prediction step*:

$$
\begin{aligned}
\hat{x}(k+1)^- &= A(k)\hat{x}(k) + B(k)u(k) \\
P(k+1)^- &= A(k)P(k)A(k)^T + G(k)Q(k)G(k)^T
\end{aligned}
$$

- *Update step*:

$$
\begin{aligned}
S(k+1) &= H(k+1)P(k+1)^- H(k+1)^T + R(k+1) \\
W(k+1) &= P(k+1)^- H(k+1)^T S(k+1)^{-1} \\
\hat{x}(k+1) &= \hat{x}(k+1)^- + W(k+1)\left(z(k+1) - H(k+1)\hat{x}(k+1)^-\right) \\
P(k+1) &= (I - W(k+1)H(k+1))P(k+1)^-
\end{aligned}
$$

Toy example

Kalman filter

Therefore:

- *Prediction step*:

$$\hat{x}_i(k+1)^- = \hat{x}_i(k) + a_i d_i(k)$$
$$P_i(k+1)^- = P_i(k) + b_i^2 \sigma_i^2$$

- *Update step* (every n_i steps):

$$S_i(k+1) = P_i(k+1)^- + \xi_i^2$$
$$W_i(k+1) = \frac{P_i(k+1)^-}{S_i(k+1)}$$
$$\hat{x}_i(k+1) = \hat{x}_i(k+1)^- + W_i(k+1)\left(z_i(k+1) - \hat{x}_i(k+1)^-\right)$$
$$P_i(k+1) = (1 - W_i(k+1))P_i(k+1)^-$$

Toy example

Kalman filter

Simplifying the terms:

- *Prediction step*:

$$\hat{x}_i(k+1)^- = \hat{x}_i(k) + a_i d_i(k)$$
$$P_i(k+1)^- = P_i(k) + b_i^2 \sigma_i^2$$

- *Update step* (every n_i steps):

$$\hat{x}_i(k+1) = \hat{x}_i(k+1)^- + \frac{P_i(k+1)^-}{P_i(k+1)^- + \xi_i^2}\left(z_i(k+1) - \hat{x}_i(k+1)^-\right)$$
$$P_i(k+1) = \frac{\xi_i^2}{P_i(k+1)^- + \xi_i^2} P_i(k+1)^-$$

Toy example

Kalman filter

From the update equations,

$$\hat{x}_i(k+1) = \hat{x}_i(k+1)^- + \frac{P_i(k+1)^-}{P_i(k+1)^- + \xi_i^2}\left(z_i(k+1) - \hat{x}_i(k+1)^-\right)$$
$$P_i(k+1) = \frac{\xi_i^2}{P_i(k+1)^- + \xi_i^2} P_i(k+1)^-$$

it is evident that:

- Whenever the sensor readings are very accurate compared to the available *a-priori knowledge*, i.e. $\xi_i^2 << P_i(k+1)^-$, we have:

$$\hat{x}_i(k+1) \approx z_i(k+1)$$
$$P_i(k+1) \approx \xi_i^2$$

Toy example

Kalman filter

Again, from the update equations,

$$\hat{x}_i(k+1) = \hat{x}_i(k+1)^- + \frac{P_i(k+1)^-}{P_i(k+1)^- + \xi_i^2}\left(z_i(k+1) - \hat{x}_i(k+1)^-\right)$$

$$P_i(k+1) = \frac{\xi_i^2}{P_i(k+1)^- + \xi_i^2} P_i(k+1)^-$$

it is evident that:

- Whenever the *a-priori knowledge* is very accurate compared to the sensor readings, i.e. $\xi_i^2 >> P_i(k+1)^-$, we have:

$$\hat{x}_i(k+1) \approx \hat{x}_i(k+1)^-$$
$$P_i(k+1) \approx P_i(k+1)^-$$

- *WARNING!*: This condition is temporary, since sooner or later $P_i(k+1)^-$ becomes comparable to ξ_i^2, i.e. uncertainty increases over time due to *dead reckoning* (*Matlab simulations*).

Toy example

Kalman filter

Finally notice that the steady state *estimation uncertainty* $P_i(k)$ can be computed in closed form by computing the fixed point of the following equation:

$$
\begin{aligned}
P_i(k) &= \frac{\xi_i^2}{P_i(k)^- + \xi_i^2} P_i(k)^- = \\
&= \frac{\xi_i^2}{P_i(k) + b_i^2\sigma_i^2 + \xi_i^2} \left(P_i(k) + b_i^2\sigma_i^2\right)
\end{aligned}
$$

Notice that this results hold for a multidimensional system as well, computing the solution of the *Riccati difference equation*.

Outline

Toy example

Consider the following toy example:

- There are two points x_1 and x_2 moving on a line, starting from unknown initial conditions $x_1(0)$ and $x_2(0)$ and subject to a first order discrete dynamic affected by noise:

$$x_i(k+1) = x_i(k) + a_i d_i(k) + b_i \varepsilon_i(k), \quad k = 0, 1, 2, \dots,$$

where $d_i(k)$ is the linear (known) displacement and $\varepsilon_i(k) \sim \mathcal{N}(0, \sigma_i^2)$ is the uncertainty affecting the displacement, supposed to be white.

Toy example

- Suppose that every $n_i \geq 1$ time steps, each point receives its *absolute* position measurement from an *exogenous* sensor, e.g. GPS, landmark detection (e.g. visual feature on a map or RFID position), anchor-based position, etc., with some uncertainty $\eta_i(k)$, i.e.

$$z_i(n_i k) = x_i(n_i k) + \eta_i(n_i k), \;\; k = 0, 1, 2, \ldots,$$

where $\eta_i(k) \sim \mathcal{N}(0, \xi_i^2)$ is a white Gaussian process.

Toy example

- Moreover, suppose that every $m_i \geq 1$ time steps, the i-th point measures the *relative position* w.r.t. $x_j(k)$ from an *exogenous* sensor, e.g. a LIDAR, a camera, an RGB-D sensor, etc., with a certain uncertainty

$$\Delta_{ij}(m_i k) = (x_j(m_j k) - x_i(m_i k)) + \eta_{ij}(m_i k), \;\; k = 0, 1, 2, \dots,$$

where $\eta_{ij}(m_i k) \sim \mathcal{N}(0, \xi_{ij}^2)$ is a white Gaussian process.

- Therefore, the output function is

$$z_{ij}(m_i k) = \Delta_{ij}(m_i k) - x_j(m_j k) = -x_i(m_i k) + \eta_{ij}(m_i k), \;\; k = 0, 1, \dots$$

- The overall uncertainty affecting this measurement is then given by $\eta_{ij}(m_i k) + P_j(m_i k)^-$.
- *WARNING!*: we are now implicitly assuming that the agent j is able to send $x_j(m_j k)$ and $P_j(m_i k)$ to the i–th agent.

Toy example

This is the essence of *collaborative localisation*!

Toy example

Kalman filter

Therefore, for the i–th or j–th agent:

- *Prediction step*:

$$\hat{x}_i(k+1)^- = \hat{x}_i(k) + a_i d_i(k)$$
$$P_i(k+1)^- = P_i(k) + b_i^2 \sigma_i^2$$

- *Update step only absolute* (every n_i steps, with $n_i k \neq m_i k$):

$$S_i(k+1) = P_i(k+1)^- + \xi_i^2$$
$$W_i(k+1) = \frac{P_i(k+1)^-}{S_i(k+1)}$$
$$\hat{x}_i(k+1) = \hat{x}_i(k+1)^- + W_i(k+1)\left(z_i(k+1) - \hat{x}_i(k+1)^-\right)$$
$$P_i(k+1) = (1 - W_i(k+1))P_i(k+1)^-$$

Toy example

Kalman filter

For only the i–th agent:

- *Update step only relative* (every m_i steps, with $n_i k \neq m_i k$):

$$S_i(k+1) = P_i(k+1)^- + \xi_{ij}^2 + P_j(k+1)^-$$
$$W_i(k+1) = \frac{P_i(k+1)^-}{S_i(k+1)}$$
$$\hat{x}_i(k+1) = \hat{x}_i(k+1)^- + W_i(k+1)\left(z_{ij}(k+1) - (-\hat{x}_i(k+1)^-)\right)$$
$$P_i(k+1) = (1 - W_i(k+1))P_i(k+1)^-$$

Toy example

Kalman filter

Finally, for the i–th agent:

- *Update step with absolute and relative* (when $n_i k = m_i k$):

$$
\begin{aligned}
S(k+1) &= H_i(k+1)P_i(k+1)^- H_i(k+1)^T + R_i(k+1)\\
W(k+1) &= P_i(k+1)^- H_i(k+1)^T S(k+1)^{-1}\\
\hat{x}_i(k+1) &= \hat{x}_i(k+1)^- + W(k+1)\left(\bar{z}(k+1) - H_i(k+1)\hat{x}_i(k+1)^-\right)\\
P_i(k+1) &= (1 - W(k+1)H_i(k+1))P_i(k+1)^-
\end{aligned}
$$

  where $\bar{z}(k+1) = [z_i(k+1), z_{ij}(k+1)]^T$, $H_i(k+1) = [1, -1]^T$ and $R_i(k+1) = \text{diag}([\xi_i^2, \xi_{ij}^2 + P_j(m_i k)^-])$.

- It is possible to show that, in the spirit of the *Fisher matrix*, using also the relative measure *the uncertainty decreases* (*Matlab simulations*).

Toy example

It is then reasonable that this idea remains valid if *also* the j–th agent measure the i–th agent every m_j time steps (*Matlab simulations*).
Unfortunately this is not always the case...
So, what is the problem?

Toy example

Consistency

The problem can be easily understood by rewriting the complete equations as it would be a centralised Kalman filter:

- Initial conditions: $\hat{x}(0) = [\hat{x}_i(0), \hat{x}_j(0)]^T$ and $P(0) = \text{diag}(P_i(0), P_j(0))$.
- *Prediction step*:

$$\hat{x}(k+1)^- = \begin{bmatrix} \hat{x}_i(k) \\ \hat{x}_j(k) \end{bmatrix} + \begin{bmatrix} a_i & 0 \\ 0 & a_j \end{bmatrix} \begin{bmatrix} d_i(k) \\ d_j(k) \end{bmatrix}$$
$$P(k+1)^- = \begin{bmatrix} P_i(k) & 0 \\ 0 & P_j(k) \end{bmatrix} + \begin{bmatrix} b_i^2\sigma_i^2 & 0 \\ 0 & b_j^2\sigma_j^2 \end{bmatrix}$$

- Notice that after the prediction step nothing different happens.

Toy example

Consistency

- *Update step*: We model that j–th agent *does not* updates its estimate, but the i–th *does* with only the relative measure from the j–th:

$$
\begin{aligned}
S(k+1) &= H_{ij}(k+1)P(k+1)^- H_{ij}(k+1)^T + R_{ij}(k+1) = \\
&= \begin{bmatrix} -1 & 1 \end{bmatrix} P(k+1)^- \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \xi_{ij}^2 = \\
&= P_i(k+1)^- + P_j(k+1)^- + \xi_{ij}^2 \\
W(k+1) &= P(k+1)^- H_{ij}(k+1)^T S(k+1)^{-1} = \\
&= \frac{1}{P_i(k+1)^- + P_j(k+1)^- + \xi_{ij}^2} \begin{bmatrix} -P_i(k+1)^- \\ 0 \end{bmatrix}
\end{aligned}
$$

- So we have to force a zero in the second element of the gain matrix of W.

Toy example

Consistency

- *Update step*: Moreover:

$$\hat{x}(k+1) = \hat{x}(k+1)^- + W(k+1)\left(z_{ij}(k+1) - H_{ij}(k+1)\hat{x}(k+1)^-\right)$$

$$\begin{aligned}
P(k+1) &= (I - W(k+1)H_{ij}(k+1))P(k+1)^- = \\
&= \frac{1}{P_i(k+1)^- + P_j(k+1)^- + \xi_{ij}^2} \cdot \\
&\cdot \begin{bmatrix} P_j(k+1)^- + \xi_{ij}^2 & P_i(k+1)^- \\ 0 & P_i(k+1)^- + P_j(k+1)^- + \xi_{ij}^2 \end{bmatrix} \cdot \\
&\cdot \begin{bmatrix} P_i(k+1)^- & 0 \\ 0 & P_j(k+1)^- \end{bmatrix}
\end{aligned}$$

- So the random variable x_i is *correlated* to the random variable x_j!

Toy example

Consistency

- As a consequence, the first time that x_j updates its position with a relative measure from x_i, it will consider the measurement H_{ji} as independent but *this is not the case*!
- This leads to the Kalman filter *inconsistency* in covariances, i.e. *the covariance decreases even if it should not*!
- Indeed...

Toy example

Consistency

- *Full update step*: Both are updated with the relative measure of the i–th agent w.r.t. the j–th agent:

$$
\begin{aligned}
S(k+1) &= H_{ij}(k+1)P(k+1)^{-}H_{ij}(k+1)^T + R_{ij}(k+1) = \\
&= \begin{bmatrix} -1 & 1 \end{bmatrix} P(k+1)^{-} \begin{bmatrix} -1 \\ 1 \end{bmatrix} + \xi_{ij}^2 = \\
&= P_i(k+1)^{-} + P_j(k+1)^{-} + \xi_{ij}^2 \\
W(k+1) &= P(k+1)^{-}H_{ij}(k+1)^T S(k+1)^{-1} = \\
&= \frac{1}{P_i(k+1)^{-} + P_j(k+1)^{-} + \xi_{ij}^2} \begin{bmatrix} -P_i(k+1)^{-} \\ P_j(k+1)^{-} \end{bmatrix}
\end{aligned}
$$

Toy example

Consistency

- *Update step*: Moreover:

$$
\begin{aligned}
\hat{x}(k+1) &= \hat{x}(k+1)^- + W(k+1)\left(z_{ij}(k+1) - H_{ij}(k+1)\hat{x}(k+1)^-\right) \\
P(k+1) &= (I - W(k+1)H_{ij}(k+1))P(k+1)^- = \\
&= \frac{1}{P_i(k+1)^- + P_j(k+1)^- + \xi_{ij}^2} \cdot \\
&\cdot \begin{bmatrix} P_j(k+1)^- + \xi_{ij}^2 & P_i(k+1)^- \\ P_j(k+1)^- & P_i(k+1)^- + \xi_{ij}^2 \end{bmatrix} \cdot \\
&\cdot \begin{bmatrix} P_i(k+1)^- & 0 \\ 0 & P_j(k+1)^- \end{bmatrix}
\end{aligned}
$$

- There is a sub-space that is not *observable*, i.e. $x_i + x_j$, hence the associated eigenvalue *grows unbounded* (i.e. *dead reckoning*).

Toy example

Solution

- To overcome this problem, a correct *covariance update* should be considered among the agents.
- This solution for a team of mobile robots can be found in: Solmaz S. Kia, Stephen Rounds, and Sonia Martínez, “Cooperative Localization for Mobile Agents: A recursive decentralized algorithm based on Kalman-filter decoupling”, IEEE Control Systems Magazine, April 2016.