Target Tracking: Computer Exercise 5

Due: 11.02.2011, 23:59

This last exercise of the TT course will be about multi-sensor issues. We are especially going to concentrate on the consistency of several track fusion methods. To this end, we need a method to evaluate the consistency of a state estimator. How to do this is going to be illustrated in the instrumental first part of this exercise. In the second part, which is the main issue of this exercise, different track fusion methods are going to be evaluated in terms of consistency on a specific example.

1 Consistency

In this first part, you are going to learn how to measure the consistency of a filter. This type of consistency measuring is then going to be used in the next part about track fusion.

The consistency issue is important for every estimation problem. In the static case (like parameter estimation), the consistency of an estimator means that when the amount of data tends to infinity, the estimate converges to the correct value of the quantity estimated. In dynamic estimation theory, the quantity to be estimated (i.e., the state) can change in time. Moreover, in online applications like TT, the amount of data about a state vector at a single time instant is far from being infinite. Hence a different consistency definition is required. We here define a consistent estimator as **an estimator whose calculated error statistics is in accordance with the true error statistics.** We can illustrate this definition with a simple example. Suppose we have an estimator which calculates an estimate $\hat{x}_{k|k}$ and a covariance for it shown as $P_{k|k}$. We say that this estimator is consistent, if

$$P_{k|k} = E_0\left[(x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k})^T | y_{0:k}\right] \tag{1}$$

where $E_0[\cdot]$ denotes the expectation operator using the **true** model parameters[1]. Here, a common fallacy is to think that this equality is always satisfied by definition if the estimator calculates $P_{k|k}$ as

$$P_{k|k} = E\left[(x_k - \hat{x}_{k|k})(x_k - \hat{x}_{k|k})^T | y_{0:k}\right] \tag{2}$$

which is the case in e.g. a KF. This is wrong because the estimator calculates the expectation (2) with **filter used** parameters which might be different than the truth. Hence the distinction between the operations $E_0[\cdot]$ and $E[\cdot]$ must be emphasized. In the following steps, you are going to learn how to measure the consistency of the KF with Chi-square statistics.

a-) True Target Data Generation: In this part, you are going to create your target's true data. For this purpose consider the model

$$x_k \triangleq \begin{bmatrix} p_k^{\mathsf{x}} & p_k^{\mathsf{y}} & v_k^{\mathsf{x}} & v_k^{\mathsf{y}} \end{bmatrix}^T = \begin{bmatrix} I_2 & TI_2 \\ 0 & I_2 \end{bmatrix} x_{k-1} + \begin{bmatrix} \frac{T^2}{2} I_2 \\ TI_2 \end{bmatrix} a_k \tag{3}$$

[1] More correctly, it denotes the expectation using the true state statistics (or equivalently the true state probability densities) because the true state process might not even satisfy the filter used model for any parameter value.

where I_2 represents 2×2 size identity matrix. We will use $T = 1\,\mathrm{s}$. The initial state x_0 should be distributed $x_0 \sim \mathcal{N}(x_0; \bar{x}_0, P_0)$ where the mean $\bar{x}_0$ and the covariance P_0 are given by

$$\bar{x}_0 \triangleq \begin{bmatrix} 5\,\mathrm{km} & 5\,\mathrm{km} & 25\,\mathrm{m/s} & 25\,\mathrm{m/s} \end{bmatrix}^T \qquad P_0 = \mathrm{diag}\left[\frac{\bar{x}_0}{10} .* \frac{\bar{x}_0}{10}\right] \tag{4}$$

Each element of 2×1 size acceleration noise a_k is distributed with $\mathcal{N}(0, \sigma_a^2)$ where $\sigma_a = 2\,\mathrm{m/s}^2$. Generate 99 seconds of the state trajectory for this target i.e., $\{x_k\}_{k=0}^{99}$.

b-) Measurement Generation: We measure the target position with a measurement standard deviation of $20\,\mathrm{m}$ for both $\mathtt{x}$ and $\mathtt{y}$ components. We assume that $P_D = 1$ and no clutter exists. Generate the target measurements i.e., $\{y_k\}_{k=0}^{99}$.

c-) Kalman Filtering: Using the true model parameters, implement a KF and obtain the state estimates and covariances, i.e., $\{\hat{x}_{k|k}, P_{k|k}\}_{k=0}^{99}$.

d-) Error Statistics: Calculate the normalized estimation error squares (NEESs) defined as

$$\epsilon_k \triangleq (x_k - \hat{x}_{k|k})^T P_{k|k}^{-1} (x_k - \hat{x}_{k|k}) \tag{5}$$

for $k = 0, \ldots, 99$. These quantities will form the basis of our consistency test.

e-) Monte Carlo Run: Do all of the above operations $N_{mc} = 100$ times. We will denote the quantities of each run with parenthesized superscripts (i), for $i = 1, \ldots, N_{mc}$. Now find the average NEESs $\{\bar{\epsilon}_k\}_{k=0}^{99}$ by taking the mean over the Monte Carlo runs as

$$\bar{\epsilon}_k = \frac{1}{N_{mc}} \sum_{i=1}^{N_{mc}} \epsilon_k^{(i)} \tag{6}$$

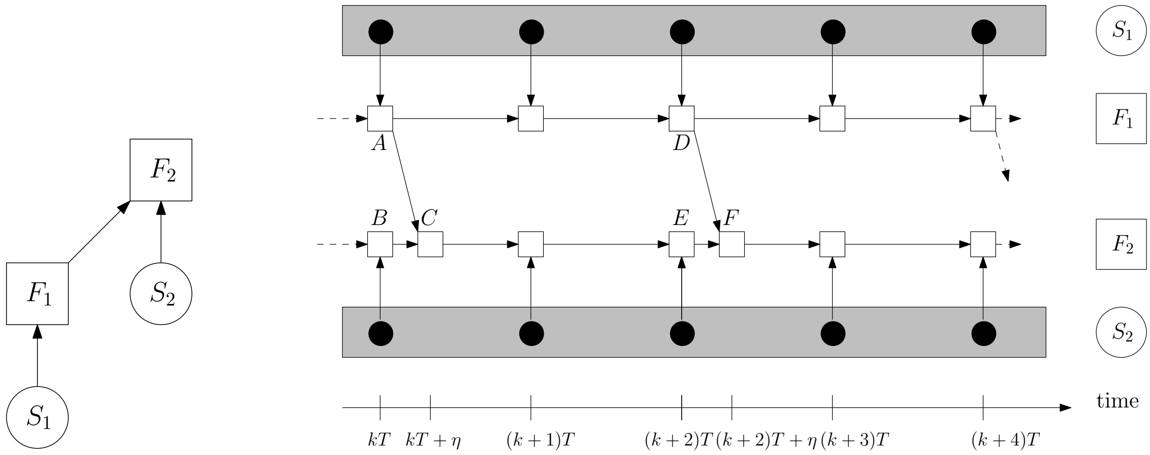
Plot the scalar quantities $\bar{\epsilon}_k$ with respect to time k.

f-) Confidence Bounds: The error statistic NEES, as should hopefully be obvious to you by now, is χ^2 distributed with degrees of freedom $n_x = 4$. Then, since our MC runs are independent from each other, the sum $\sum_{i=1}^{N_{mc}} \epsilon_k^{(i)} = N_{mc}\bar{\epsilon}_k$ is χ^2 distributed with degrees of freedom $N_{mc}n_x = 400$. One then can find lower and upper thresholds for the sum $\sum_{i=1}^{N_{mc}} \epsilon_k^{(i)} = N_{mc}\bar{\epsilon}_k$ as

$$\gamma_{\min} = \mathtt{chi2inv}(0.005, N_{mc}n_x) \quad \text{and} \quad \gamma_{\max} = \mathtt{chi2inv}(1 - 0.005, N_{mc}n_x) \tag{7}$$

which ensures that $P(\gamma_{\min} \leq \sum_{i=1}^{N_{mc}} \epsilon_k^{(i)} \leq \gamma_{\max}) = 0.99$. Hence we can ensure $P(\bar{\epsilon}_{\min} \leq \bar{\epsilon}_k \leq \bar{\epsilon}_{\max}) = 0.99$ by defining $\bar{\epsilon}_{\min} \triangleq \gamma_{\min}/N_{mc}$ and $\bar{\epsilon}_{\max} \triangleq \gamma_{\max}/N_{mc}$. Plot the thresholds $\bar{\epsilon}_{\min}$ and $\bar{\epsilon}_{\max}$ on top of $\bar{\epsilon}_k$ curves and check whether your KF is consistent at all times. Note that the plotted $\bar{\epsilon}_k$ values might go out of the bounds you found at about 1% of the times on average (because we found those bounds using 0.99 probabilities).

g-) Different Cases: Now replace the filter used process noise standard deviation (but not the true process noise standard deviation) to half of the true value. What happens to the statistics? What about two times the true value? What happens when you do the same with the filter used measurement noise standard deviation?



(a) (b)

Figure 1: Fusion architecture (a) and information graph (b) for the problem investigated in the exercise. We are going to use $\eta \approx 0$, i.e., the information transmission is immediate, in our experiments.

2 Track Fusion

This second problem of CE-5 is about different track fusion methods. We are going to consider the fusion architecture shown in Figure 1(a). The information graph of the fusion problem we consider is given in Figure 1(b). In this scenario, we have two local agents called as F_1 and F_2 connected to sensors S_1 and S_2 respectively. The local agents run their local trackers (KFs) based on their local sensor information. We assume that both local agents sensors are synchronized and obtain their measurements at time instants kT where $T = 1$ seconds for $k = 0, \ldots, 99$. The local agents are connected to each other in an hierarchical way in that F_1 periodically sends its local estimate $\hat{x}^1_{k|k}$ and covariance $P^1_{k|k}$ to F_2. This communication takes some time shown with η in the information graph of Figure 1(b). For the sake of simplicity, we assume in this exercise that this communication is instantaneous, i.e., $\eta = 0$. The communication period is selected to be 2 seconds, i.e., at the instants $k = 0 : 2 : 99$. When local agent F_2 gets the estimate $\hat{x}^1_{k|k}$ and covariance $P^1_{k|k}$, it fuses this information with its local quantities $\hat{x}^2_{k|k}$ and $P^2_{k|k}$ and replaces the local quantities $\hat{x}^2_{k|k}$ and $P^2_{k|k}$ with the fused estimate and covariance respectively. The aim of the fusion at the local agent F_2 is to obtain, at the end, estimates as close to the centralized estimates as possible. For this purpose, several track fusion algorithms will be evaluated below.

- **a-) True Target Data Generation:** This part is exactly the same as the corresponding part of the previous question.
- **b-) Measurement Generation:** The sensors S_1 and S_2 independently measure the target position with a measurement standard deviation of 20 m for both x and y components. We assume that $P_D = 1$ and no clutter exists. Generate the target measurement sets $\{y^1_k\}_{k=0}^{99}$ and $\{y^2_k\}_{k=0}^{99}$. Notice that measurement noises for the sensors should be created separately.
- **c-) Centralized Solution:** Now assume that both measurement sets $\{y^1_k\}_{k=0}^{99}$ and $\{y^2_k\}_{k=0}^{99}$ are available at a fictitious fusion center. Obtain the centralized estimates using a matched Kalman filter (to the true model). This can be done by stacking y^1_k and y^2_k into an aug-

mented measurement vector z_k and running a Kalman filter with corresponding augmented C matrix and block diagonal measurement noise covariance.

d-) **Decentralized Solution:** Obtain the estimates and covariances $\{\hat{x}^2_{k|k}, P^2_{k|k}\}_{k=0}^{99}$ (of the local agent F_2) of the fusion scenario depicted in Figure 1. The fusion of local estimates is to be done with the so-called naive fusion (independence assumption) illustrated in Slide 25 of Lecture 6. Summarizing the main idea: At time instants $k = 0 : 2 : 99$ we replace the local quantities $\hat{x}^2_{k|k}$, $P^2_{k|k}$ with their fused equivalents obtained as

$$(P^f_{k|k})^{-1} =(P^1_{k|k})^{-1} + (P^2_{k|k})^{-1} \tag{8}$$

$$(P^f_{k|k})^{-1}\hat{x}^f_{k|k} =(P^1_{k|k})^{-1}\hat{x}^1_{k|k} + (P^2_{k|k})^{-1}\hat{x}^2_{k|k} \tag{9}$$

where the quantities with superscript f denote the fused ones. This operation is shown on the information graph of Figure 1(b) with the estimate C (or F) obtained by fusing estimates at A (or D) and B (or F).

e-) **Channel Filter:** Repeat part **d** by using channel filter instead of naive fusion. Channel filter is illustrated in Slide 27 of Lecture 6. Summarizing the main idea: We modify the independent fusion formulas in part **d** by subtracting the extrapolated (predicted) received quantities (common information) obtained at the last fusion time, i.e.,

$$(P^f_{k|k})^{-1} =(P^1_{k|k})^{-1} + (P^2_{k|k})^{-1} - (P^1_{k|k-2})^{-1} \tag{10}$$

$$(P^f_{k|k})^{-1}\hat{x}^f_{k|k} =(P^1_{k|k})^{-1}\hat{x}^1_{k|k} + (P^2_{k|k})^{-1}\hat{x}^2_{k|k} - (P^1_{k|k-2})^{-1}\hat{x}^1_{k|k-2} \tag{11}$$

On the information graph of Figure 1(b), this corresponds to obtaining estimate F by fusing estimates D and E and then defusing (subtracting) the predicted version of estimate A which is the last communication in the *channel* between F_1 and F_2. At the first fusion instant ($k = 0$), since there had not been any communication, naive fusion formulas should be used. The extrapolated quantities $\hat{x}^1_{k|k-2}$ and $P^1_{k|k-2}$ can be calculated by making two consecutive prediction (time) updates to the quantities $\hat{x}^1_{k-2|k-2}$ and $P^1_{k-2|k-2}$ respectively.

f-) **Correlation Independent Fusion:** Repeat part **d** with either of largest ellipsoid algorithm (LEA) or covariance intersection (CI) fusion. The information about these methods can be found in Slides 29-31 of Lecture 6. The implementation of LEA is summarized in Slides 29-30 of Lecture 6. If you choose to implement CI instead, you can just discretize the optimization parameter $w \in [0, 1]$ with e.g. 20 values and choose the best one instead of a continuous optimization routine.

g-) **Evaluation:** Make at least 100 MC runs using different realizations of the true target trajectories and measurements at each run with the above algorithms. Evaluate the considered track fusion methods (including the centralized results) in terms of consistency (first part of this exercise) and RMS position and velocity errors. Plot the NEES curves with confidence boundaries (as in the first part of this exercise) and RMS error curves for this purpose (with respect to time). Note that evaluations should be done only on the F_2 estimates and covariances. Comment on your findings.