A Brief Tutorial On Recursive Estimation With Examples From Intelligent Vehicle Applications (Part II): System Models

Hao Li

A Brief Tutorial On Recursive Estimation With Examples From Intelligent Vehicle Applications (Part II): System Models

Hao Li

Abstract

In the first article of the series "A brief tutorial on recursive estimation", we have given a sketch on the overview of recursive estimation, with examples from intelligent vehicle applications. In this article, we focus rather on a "local" issue, i.e. the **system model**, and show the importance of a suitable system model for an estimation process.

Keywords: recursive estimation, state, system model, Kalman filter (KF), intelligent vehicles

1 Introduction

This article follows the first article [1] of the series "A brief tutorial on recursive estimation", in which we have reviewed several basic concepts concerning estimation and explained the generic spirit of recursive estimation using Bayesian inference. In this article, we focus on the issue of **system model**.

As we have introduced in [1], for a state, a system model is defined as a (mathematical) model describing the laws (physical, chemical, biological etc) that govern the evolution or change of the state. In fact, additional explanations should be given to clarify this definition.

Before continuing, we would remind readers of the difference between the world (or the nature, the universe etc) existing objectively there and the world in our knowledge. Here, we simply call the former the **objective world** and call the latter the **image world**—Along thousands of years of the development of human civilization, many philosophers have reflected

on the relationship between the objective world and the image world. The basic difference in their points of view can be summarized, if expressed in mathematical terms, as one point: they calculated the correlation between the objective world and the image world differently. For example, for *Lao Zi* (the creator of *Taoism* or "*Dao Jiao*") and many scientists, this correlation can asymptotically converge to 1 as the human strives to know more and more about the objective world. For *Buddha*, this correlation can never be more than 0, because the objective world is totally "empty" in his opinion. For some others, this correlation may achieve a certain value but not 1 because certain "God" would not allow the human to know too much.

Here we have no intention to distract readers with stories and opinions of this or that philosopher. The efforts of all those clever philosophers tell us a fact, i.e. what we indeed try to highlight with above philosopher examples: **system modeling** in an absolutely objective way is difficult—it is even difficult to represent objectively the entities that we try to examine, not to say to describe objectively the evolution of these entities.

In practice, we need **approximation** for system modeling. First, we need to choose a state to approximately characterize an entity that we care about. Then we need to establish a system model in terms of the chosen state to approximately describe the evolution of the entity. For the same entity, the system model may be established (including the choice of the state itself) in different ways, depending on how we approximate the entity and its evolution.

To have desirable estimation of a property of the entity, establishing a suitable system model is an important preliminary task. We stop abstract explanations here and continue our explanations with concrete examples in following sections.

2 Examples: Vehicle Localization (1D)

In many vehicle localization applications such as presented in [2] [3] [4] [5] [6], the vehicle pose is treated as the vehicle state $\mathbf{x}$, whereas the vehicle speed, yawrate etc are measurable and treated explicitly as the system input $\mathbf{u}$. Here, we also consider the example of vehicle localization in a 1D case presented in [1], where the vehicle position p is treated as the state and the vehicle speed v is treated as the system input. The system model is given as

the following 1D kinematic model:

$$p_t = p_{t-1} + v_t \Delta T \tag{1}$$

Given current vehicle position and vehicle speed, the vehicle position in next period can be predicted via (1). This reflects an important characteristic of system models, i.e. **deterministic**—the "deterministic" here is in **probabilistic** sense: the uncertainty of the state prediction depends only on the uncertainty of current state and system input. In other words, if there is no error in current state estimate and system input measurement, and if there is no system model error, then the state in next period can be precisely predicted by the system model.

2.1 System modeling in different ways

Now imagine that we do not have measurements on the vehicle speed and we only have measurements on the vehicle position. We still want to carry out estimation on the vehicle position. In this case, the system model (1) is no longer suitable for this estimation task, because it has an undetermined variable and can not satisfy the requirement of being deterministic (in probabilistic sense)—As we can see here, for the same objective entity (the vehicle) and its evolution (its motion), a system model may be suitable in some cases and may not in other cases. So *the appropriateness of a system model is not absolute but depends on concrete system configurations*.

To handle the problem that the system model (1) is no longer deterministic (in probabilistic sense), we have to resort to certain ***a priori* knowledge** on the vehicle to establish a deterministic (in probabilistic sense) system model for its motion.

Constant position (CP) model: If we have *a priori* knowledge that the vehicle is stationary, we may establish a constant position (CP) model to describe its motion. In this case, the system model is formulated as:

$$p_t = p_{t-1} + \Delta p_t \tag{2}$$

where Δp_t represents the vehicle position variation which may be treated as a random noise and may be assumed to follow the Gaussian distribution $N(0, \Sigma_p)$. The covariance Σ_p may be set according to our experience, depending on the degree of our certitude on the *a priori* knowledge.

Constant velocity (CV) model: If we have *a priori* knowledge that the vehicle moves at constant velocity, we may establish a constant velocity (CV) kinematic model to describe its motion and the system model is formulated as:

$$\begin{bmatrix} p_t \\ v_t \end{bmatrix} = \begin{bmatrix} 1 & \Delta T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_{t-1} \\ v_{t-1} \end{bmatrix} + \begin{bmatrix} 0 \\ \Delta v_t \end{bmatrix} \tag{3}$$

where Δv_t represents the vehicle velocity variation which may be treated as a random noise and may be assumed to follow the Gaussian distribution $N(0, \Sigma_v)$. The system model (3) may be understood as follows: imagine the vehicle has been moving at constant velocity v_{t-1} during the current system period and at the last instant of this period the vehicle velocity suddenly jumps to v_t with the difference Δv_t following the Gaussian distribution $N(0, \Sigma_v)$.

Constant acceleration (CA) model: if we have *a priori* knowledge that the vehicle moves at constant acceleration, we may establish a constant acceleration (CA) kinematic model to describe its motion and the system model is formulated as:

$$\begin{bmatrix} p_t \\ v_t \\ a_t \end{bmatrix} = \begin{bmatrix} 1 & \Delta T & \Delta T^2/2 \\ 0 & 1 & \Delta T \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{t-1} \\ v_{t-1} \\ a_{t-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \Delta a_t \end{bmatrix} \tag{4}$$

where Δa_t represents the vehicle acceleration variation which may be treated as a random noise and may be assumed to follow the Gaussian distribution $N(0, \Sigma_a)$. The system model (4) may be understood as follows: imagine the vehicle has been moving at constant acceleration during the current system period and at the last instant of this period the vehicle acceleration suddenly jumps to a_t with the difference Δa_t following the Gaussian distribution $N(0, \Sigma_a)$.

2.2 Application of the Kalman filter using the CP, CV, and CA models

After introducing the CP, CV, and CA models, we return to the problem of estimating the vehicle position given a sequence of raw position measurements. We still use the Kalman filter (KF) [7] whose essence has been explained in the previous tutorial part [1].

Denote the vehicle position measurement as z; denote the measurement error as γ, which is assumed to follow the Gaussian distribution with zero mean and the covariance Σ_γ, i.e. $\gamma \sim N(0, \Sigma_\gamma)$. If we use the CP model as the system model, then the measurement model is given as:

$$z_t = p_t + \gamma_t \tag{5}$$

If we use the CV model as the system model, then the measurement model is given as:

$$z_t = \mathbf{H_{cv}} \begin{bmatrix} p_t \\ v_t \end{bmatrix} + \gamma_t \tag{6}$$

$$\mathbf{H_{cv}} = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

If we use the CA model as the system model, then the measurement model is given as:

$$z_t = \mathbf{H_{ca}} \begin{bmatrix} p_t \\ v_t \\ a_t \end{bmatrix} + \gamma_t \tag{7}$$

$$\mathbf{H_{ca}} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$$

Concrete formulas of the KF using the CP model, using the CV model, and using the CA model are given respectively as follows:

2.2.1 The KF using the CP model

Prediction:

$$\bar{p}_t = \hat{p}_{t-1}$$
$$\bar{\Sigma}_t = \hat{\Sigma}_{t-1} + \Sigma_p$$

Update:

$$K = \bar{\Sigma}_t(\bar{\Sigma}_t + \Sigma_\gamma)^{-1}$$
$$\hat{p}_t = \bar{p}_t + K(z_t - \bar{p}_t)$$
$$\hat{\Sigma}_t = (1 - K)\bar{\Sigma}_t$$

2.2.2 The KF using the CV model

Prediction:

$$\begin{bmatrix}\bar{p}_t\\ \bar{v}_t\end{bmatrix} = \begin{bmatrix}1 & \Delta T\\ 0 & 1\end{bmatrix}\begin{bmatrix}\hat{p}_{t-1}\\ \hat{v}_{t-1}\end{bmatrix}$$
$$\bar{\mathbf{\Sigma}}_t = \begin{bmatrix}1 & \Delta T\\ 0 & 1\end{bmatrix}\hat{\mathbf{\Sigma}}_{t-1}\begin{bmatrix}1 & 0\\ \Delta T & 1\end{bmatrix} + \begin{bmatrix}0 & 0\\ 0 & \Sigma_v\end{bmatrix}$$

Update:

$$\mathbf{K} = \bar{\mathbf{\Sigma}}_t\mathbf{H_{cv}^T}(\mathbf{H_{cv}}\bar{\mathbf{\Sigma}}_t\mathbf{H_{cv}^T} + \Sigma_\gamma)^{-1}$$
$$\begin{bmatrix}\hat{p}_t\\ \hat{v}_t\end{bmatrix} = \begin{bmatrix}\bar{p}_t\\ \bar{v}_t\end{bmatrix} + \mathbf{K}(z_t - \mathbf{H_{cv}}\begin{bmatrix}\bar{p}_t\\ \bar{v}_t\end{bmatrix})$$
$$\hat{\mathbf{\Sigma}}_t = (\mathbf{I} - \mathbf{K}\mathbf{H_{cv}})\bar{\mathbf{\Sigma}}_t$$

2.2.3 The KF using the CA model

Prediction:

$$\begin{bmatrix}\bar{p}_t\\ \bar{v}_t\\ \bar{a}_t\end{bmatrix} = \begin{bmatrix}1 & \Delta T & \Delta T^2/2\\ 0 & 1 & \Delta T\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\hat{p}_{t-1}\\ \hat{v}_{t-1}\\ \hat{a}_{t-1}\end{bmatrix}$$
$$\bar{\mathbf{\Sigma}}_t = \begin{bmatrix}1 & \Delta T & \Delta T^2/2\\ 0 & 1 & \Delta T\\ 0 & 0 & 1\end{bmatrix}\hat{\mathbf{\Sigma}}_{t-1}\begin{bmatrix}1 & \Delta T & \Delta T^2/2\\ 0 & 1 & \Delta T\\ 0 & 0 & 1\end{bmatrix}^T + \begin{bmatrix}0 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & \Sigma_a\end{bmatrix}$$

Update:

$$\mathbf{K} = \bar{\mathbf{\Sigma}}_t\mathbf{H_{ca}^T}(\mathbf{H_{ca}}\bar{\mathbf{\Sigma}}_t\mathbf{H_{ca}^T} + \Sigma_\gamma)^{-1}$$
$$\begin{bmatrix}\hat{p}_t\\ \hat{v}_t\\ \hat{a}_t\end{bmatrix} = \begin{bmatrix}\bar{p}_t\\ \bar{v}_t\\ \bar{a}_t\end{bmatrix} + \mathbf{K}(z_t - \mathbf{H_{ca}}\begin{bmatrix}\bar{p}_t\\ \bar{v}_t\\ \bar{a}_t\end{bmatrix})$$
$$\hat{\mathbf{\Sigma}}_t = (\mathbf{I} - \mathbf{K}\mathbf{H_{ca}})\bar{\mathbf{\Sigma}}_t$$

2.3 Simulation

We test the performance of the KF using respectively the CP model, the CV model and the CA model in simulation—As presented in the previous article

[1], in this article, and potentially in further articles of this series of tutorial, we always prefer simulation to demonstrate the performance of this or that method we present. This is by no means to imply that simulation is more important than practice in real applications. On the contrary, real applications are important as they are usually the core motivation for developing estimation methods and they are the final "judge" to evaluate the feasibility of these methods. The reasons why we prefer simulation in this series of tutorial are mainly two-folds. First, the purpose of this series of tutorial is to enlighten beginners on basic spirit of recursive estimation and on characteristics of some commonly used recursive estimation methods, whereas showing the feasibility of certain estimation method in certain application is out of our focus. So we employ simulation to demonstrate the "pure" performance of the presented estimation methods, exempt from the influence of *ad hoc* implementation factors in real practice. Second, in simulation, we can carry out a fair comparative study among different methods, because in simulation we can eliminate any *ad hoc* implementation factor that may bias the estimates towards or against this or that method. Note that a fair comparative study among different candidate methods is usually a valuable guide for real practice.

2.3.1 Performance of the KF using the CP model

In the simulation, the ground-truth was synthesized according to the CP model: let $p_t \equiv 10(m)$. The position measurements were synthesized according to $z_t \sim N(p_t, \Sigma_\gamma)$; let $\Sigma_\gamma = 5.0^2(m^2)$. For the estimation process, let $\Delta T = 1(s)$; let $\Sigma_p = 0$, as we had the "*a priori* knowledge" that the vehicle was stationary. We set the initial estimate arbitrarily (say $\hat{p}_0 = 0(m)$) with an "infinite" covariance $\hat{\Sigma}_0$ to indicate our total incertitude on the initial "guess"—In programming, we can not really set an infinite value, instead we can choose to set a very large value; however, not too large, otherwise the numerical structure of the estimation procedures might be poor and numerical problems might arise.

The KF using the CP model was applied to the synthesized data and estimates on the vehicle position were obtained. The estimate errors were computed and compared with the measurement errors. The result of 100 Monte Carlo trials is shown in Fig.1.

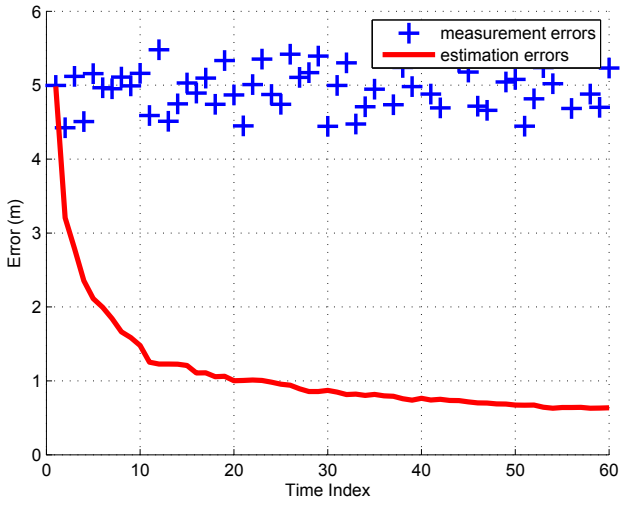

Figure 1: Position estimate and measurement errors for 100 Monte Carlo trials

2.3.2 Performance of the KF using the CV model

In the simulation, the ground-truth was synthesized according to the CV model: let $p_0 = 10(m)$ and $v_t \equiv 10(m/s)$. The position measurements were synthesized according to $z_t \sim N(p_t, \Sigma_\gamma)$; let $\Sigma_\gamma = 5^2(m^2)$. For the estimation process, let $\Delta T = 1(s)$; let $\Sigma_v = 0$, as we had the "*a priori* knowledge" that the vehicle was moving at constant velocity. We set the initial estimate arbitrarily (say $\hat{p}_0 = 0(m)$ and $\hat{v}_0 = 0(m/s)$) with an "infinite" covariance $\hat{\mathbf{\Sigma}}_0$ to indicate our total incertitude on the initial "guess".

The KF using the CV model was applied to the synthesized data and the estimates were compared with the ground-truth. The results of 100 Monte Carlo trials are shown in Fig.2 and Fig.3.

2.3.3 Performance of the KF using the CA model

In the simulation, the ground-truth was synthesized according to the CA model: let $p_0 = 10(m)$, $v_0 = 0(m/s)$, and $a_t \equiv 0.2(m/s^2)$. The position measurements were synthesized according to $z_t \sim N(p_t, \Sigma_\gamma)$; let $\Sigma_\gamma = 5^2(m^2)$. For the estimation process, let $\Delta T = 1(s)$; let $\Sigma_a = 0$, as we had the "*a priori*

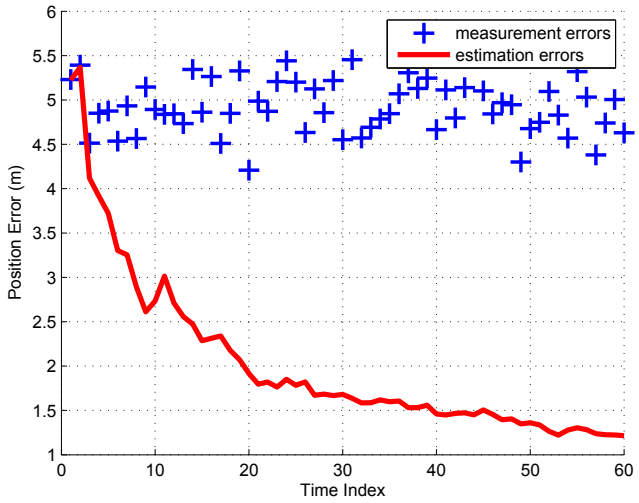

Figure 2: Position estimate and measurement errors for 100 Monte Carlo trials

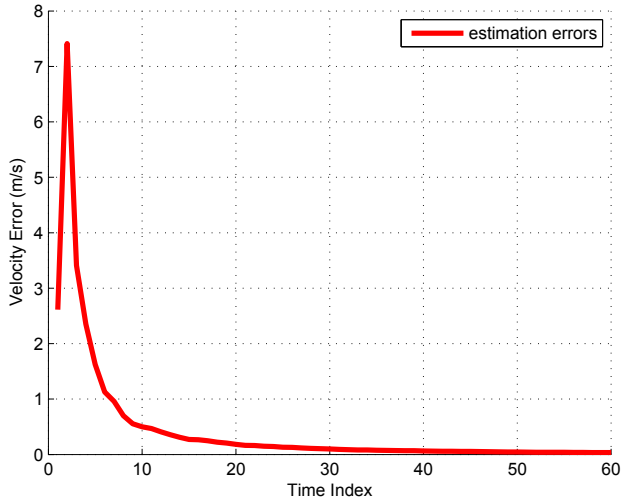

Figure 3: Velocity estimate errors for 100 Monte Carlo trials

knowledge" that the vehicle was moving at constant acceleration. We set the initial estimate arbitrarily (say $\hat{p}_0 = 0(m)$, $\hat{v}_0 = 0(m/s)$, and $\hat{a}_0 = 0(m/s^2)$) with an "infinite" covariance $\hat{\mathbf{\Sigma}}_0$ to indicate our total incertitude on the initial "guess".

The KF using the CA model was applied to the synthesized data and the estimates were compared with the ground-truth. The results of 100 Monte Carlo trials are shown in Fig.4 and Fig.5.

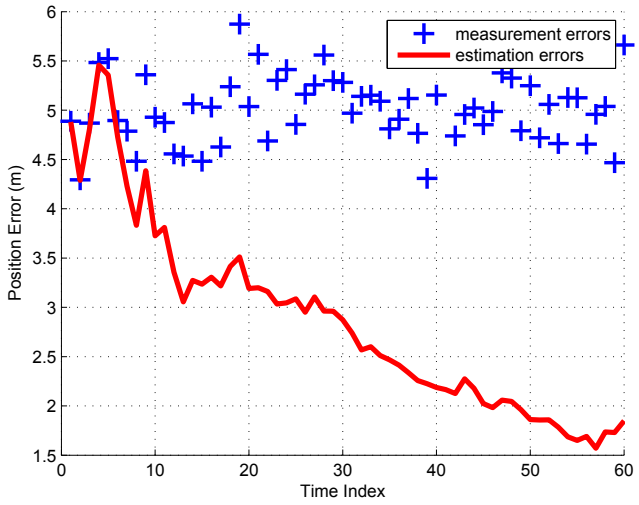

Figure 4: Position estimate and measurement errors for 100 Monte Carlo trials

2.4 Result interpretation and discussion

As we can see from results presented in previous subsections, if we have ideal *a priori* knowledge and have a suitable system model, we can have rather desirable estimates on the state. As shown in Fig.1, Fig.2, and Fig.4, the position estimate errors are considerably smaller than the raw measurement errors and decrease monotonically—Here, we have just shown the results in a limited interval of time. In fact, if we extend the time index to enough long, one will easily see the tendency that the estimate errors converge asymptotically to zero—Besides, as shown in Fig.3 and Fig.5, the vehicle velocity is well

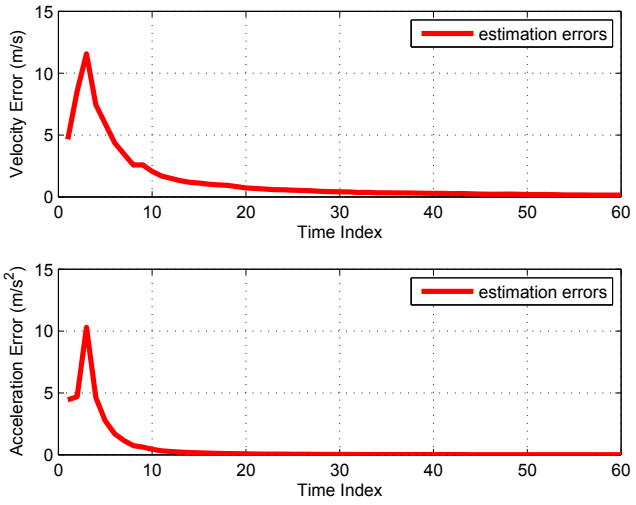

Figure 5: Velocity and acceleration estimate errors for 100 Monte Carlo trials

recovered by the estimation methods, as the velocity estimate errors also converge asymptotically to zero—Note that we only have direct measurements on the vehicle position—Above results demonstrate once again the utilities of estimation methods that we have already explained in the previous article [1], i.e. "can know" and "know better".

Estimating the full state i.e. the position, the velocity etc of an object, given measurements only on the object position, is a **tracking** (or object tracking) process [8]. In the context of intelligent vehicles, tracking of surrounding objects of a vehicle is important, especially for applications of collision avoidance [9] [10] [11] [12]. For a vehicle, to guarantee its navigation safety, it has to monitor in real time current positions of its surrounding objects; besides, it also has to predict future positions of these objects so that it can take anticipatory action to avoid potential collision with these objects. In order to achieve this, it needs to know the velocities of these objects. Perceptive sensors usually installed on an intelligent vehicle, such as laser scanners etc, can only have measurements on the object position. Therefore, the vehicle needs to estimate indirectly the velocities of its surrounding objects with measurements on the positions of these objects. In other words, the vehicle needs to track its surrounding objects.

The results presented above are desirable, it seems that we have some methods which can well fulfill the task specified at the beginning of this section, i.e. estimating the position of the vehicle. Not limited to this, we can even have accurate estimates on the vehicle velocity and the vehicle acceleration, though no information on them is directly available. In one word, "all is perfect".

Yes, we are right; "all is perfect" and this "all" also includes the scenarios and the *a priori* knowledge assumed available. First, the scenarios conceived were perfectly simple; the vehicle was set to be strictly stationary, be moving at strictly constant velocity, or be moving at strictly constant acceleration. However, this sort of ideal movement can only appear in tutorials or textbooks and can hardly happen in reality. Second, even more perfectly, we could have *a priori* knowledge that is completely consistent with the truth. However, how can this normally be possible in reality? For above examples, how can the estimator know beforehand the movement pattern of the vehicle in the next period?

What would be the performance of previously presented estimation techniques (the KF using the CP model, the KF using the CV model, and the KF using the CA model) if the *a priori* knowledge was wrong? Would we still get desirable results as those presented previously? We will examine these problems in Section 3.

3 System Model Mismatch

A system model which describes a phenomenon normally relies on certain *a priori* knowledge or assumptions on this phenomenon. The CP model, the CV model, and the CA model are examples where assumptions of constant vehicle position, of constant vehicle velocity, and of constant vehicle acceleration are followed respectively. On the other hand, there will normally exist certain inconsistency between our assumptions and the truth. Consequently, there will also exist certain inconsistency between how an entity actually evolves and what a system model describes. This inconsistency is what we call as **system model mismatch** or **model mismatch** for short. A method working perfectly in ideal cases may perform poorly in cases with system model mismatch, as will be shown soon.

3.1 Mismatch 1: low-order model vs. high-order movement

We return to the examples described in section 2. In another set of trials, we used data synthesized according to the CA model (see section 2.3.3). Instead of applying the KF using the CA model to these data, we applied the KF using the CV model. In other words, we carried out the estimation as if the vehicle was moving at constant velocity, whereas the vehicle was actually accelerating. We can notice apparent model mismatch in this estimation case. The results of 100 Monte Carlo trials are shown in Fig.6 and Fig.7.

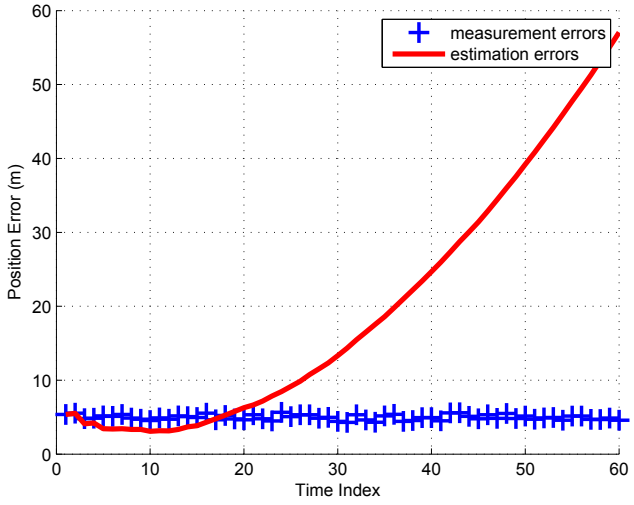

Figure 6: Position estimate and measurement errors for 100 Monte Carlo trials

As we can see in Fig.6 and Fig.7, both the position estimates and the velocity estimates diverge severely from the ground-truth. This example shows that system model mismatch can severely degrade the performance of an estimation method and even make the estimates useless.

We give some intuitive explanations on why the model mismatch i.e. the inconsistency between the CV model and the actual movement of acceleration degrades the performance of the KF using the CV model (see section 2.2.2). Recall that we set $\Sigma_v = 0$ (see section 2.3.2) as we followed strictly the

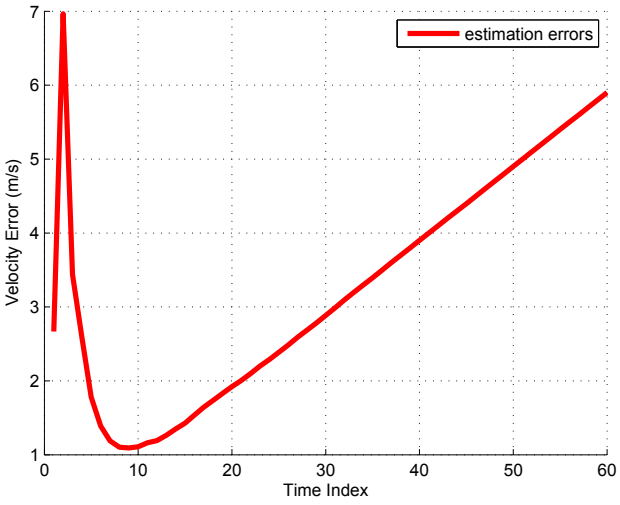

Figure 7: Velocity estimate errors for 100 Monte Carlo trials

constant velocity assumption in the tests. This means that we lay too much confidence on the predictive ability of the CV model; this over-confidence on the CV model tends to bias the estimates towards the predicted *a priori* and make it difficult for the estimates to be corrected (updated) by new measurements. As a result, the estimates diverge far and far away from the ground-truth.

As an expedient solution to handle potential velocity variation while using the CV model, one may set Σ_v empirically to certain value to represent the uncertainty of the constant velocity assumption that underlies the CV model. In other words, we use Σ_v to reflect the degree of potential mismatch of the CV model. For example, we set $\Sigma_v = 0.1^2$ and performed again the KF using the CV model on synthesized data generated according to the CA model. The results of 100 Monte Carlo trials are shown in Fig.8 and Fig.9. We set $\Sigma_v = 1.0^2$, the results of 100 Monte Carlo trials are shown in Fig.10 and Fig.11. We set $\Sigma_v = 10.0^2$, the results of 100 Monte Carlo trials are shown in Fig.12 and Fig.13.

As shown in Fig.8 and Fig.9, the estimates under $\Sigma_v = 0.1^2$ no longer diverge like the estimates under $\Sigma_v = 0.0$, yet still suffering from rather large errors. As shown in Fig.10 and Fig.11, the estimates under $\Sigma_v = 1.0^2$

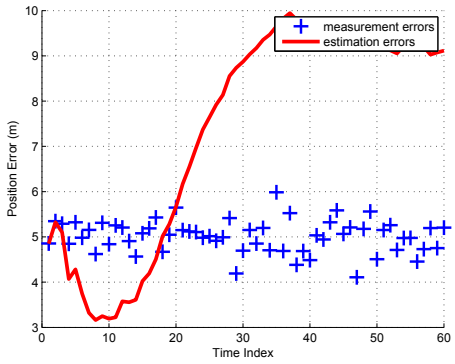

Figure 8: Position estimate and measurement errors for 100 Monte Carlo trials

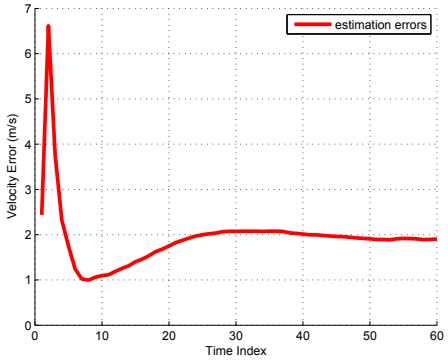

Figure 9: Velocity estimate errors for 100 Monte Carlo trials

become even better: the position estimates are apparently smaller than the raw position measurements and the velocity estimate errors become much smaller. It seems that increasing Σ_v can improve the estimation performance. However, as we further increase Σ_v to $\Sigma_v = 10.0^2$, the estimates deteriorate again, as shown in Fig.12 and Fig.13.

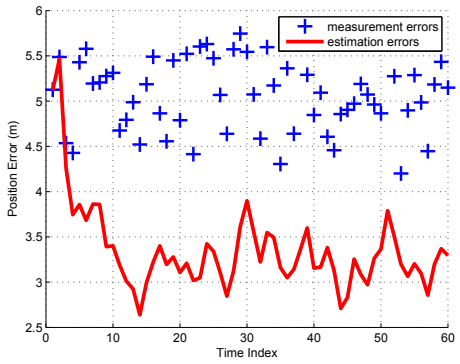

Figure 10: Position estimate and measurement errors for 100 Monte Carlo trials

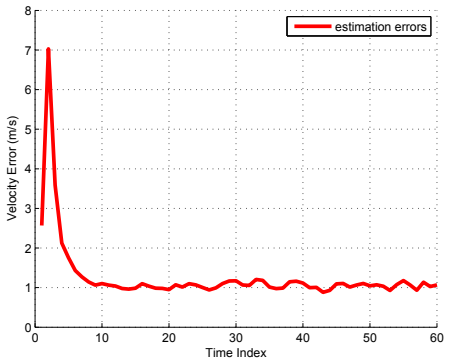

Figure 11: Velocity estimate errors for 100 Monte Carlo trials

Discussion

The tendency reflected by Fig.8 to Fig.13 seems to imply that given a movement of acceleration there would be an optimal Σ_v under which the KF using the CV model achieves best performance. On the other hand, this optimal Σ_v is not fixed for all cases. As the acceleration changes, the optimal Σ_v will also change. This causes difficulties to tuning the Σ_v in real practice if we use the CV model to handle cases of acceleration movement. More generally, it is usually difficult to tune the **process noise** (in above example, i.e. the

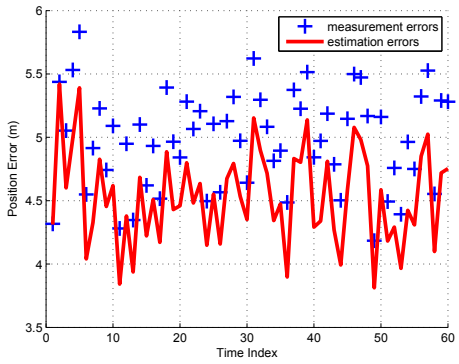

Figure 12: Position estimate and measurement errors for 100 Monte Carlo trials

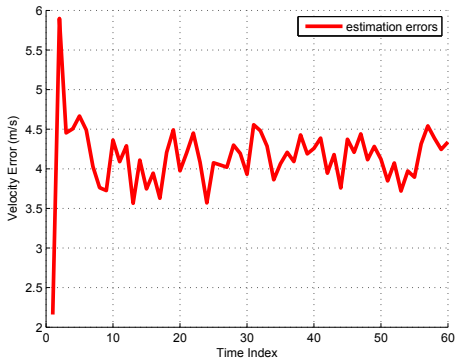

Figure 13: Velocity estimate errors for 100 Monte Carlo trials

model error part represented by Σ_v) if we use a lower-order model to handle cases of higher-order movement.

Besides, no matter how Σ_v is set in above tests, one can notice the static errors of the velocity estimates, as shown in Fig.9, Fig.11, and Fig.13. The reason for the existence of these static errors may be understood intuitively as follows: in the CV model, the velocity variation is modeled as a random noise and hence the average acceleration is still assumed to be zero. Thus in an acceleration movement, the estimator tends to favor velocity estimates with bias according to which the prediction may better "catch up with" the

actual acceleration movement on the whole.

In contrast with the results shown in Fig.9, Fig.11, and Fig.13, the velocity estimates shown in Fig.5 converge asymptotically to the ground-truth, without any static estimate error.

Above discussions tell us a point: for a higher-order movement, we had better utilize a system model matching this movement, not a lower-order system model with certain expedient adaptation.

3.2 Mismatch 2: high-order model vs. low-order movement

In the previous subsection, we have shown a kind of model mismatch i.e. using a lower-order system model to characterize a higher-order movement. In this subsection, we will show an opposite kind of model mismatch i.e. using a higher-order system model to characterize a lower-order movement.

One may think that a lower-order movement is a special case of a higher-order movement. For example, the movement of constant velocity can be seen as a special case of the movement of constant acceleration i.e. a movement of constant acceleration zero. A model being able to describe a higher-order movement can also describe a lower-order movement. So using a higher-order system model in the estimation will yield at least no worse results than using a lower-order system model.

This argument seems plausible. We do not make any judgement on this argument for the moment but check it with simulation tests first. We used data synthesized according to the CV model (see section 2.3.2). We applied both the KF using the CV model and the KF using the CA model to these synthesized data simultaneously. We set $\Sigma_v = 0$ and $\Sigma_a = 0$ because we "knew" *a priori* that the vehicle was set to be moving at constant velocity (also at constant acceleration zero). The results of 100 Monte Carlo trials are shown in Fig.14 and Fig.15.

As we can see, errors of the estimates obtained by using the CV model and by using the CA model both have the tendency of decreasing monotonically. We can also easily note that the estimate errors associated with use of the CV model decrease more quickly than those associated with use of the CA model. In other words, the KF using the CV model performs better than the KF using the CA model in the sense that it is more responsive than the latter.

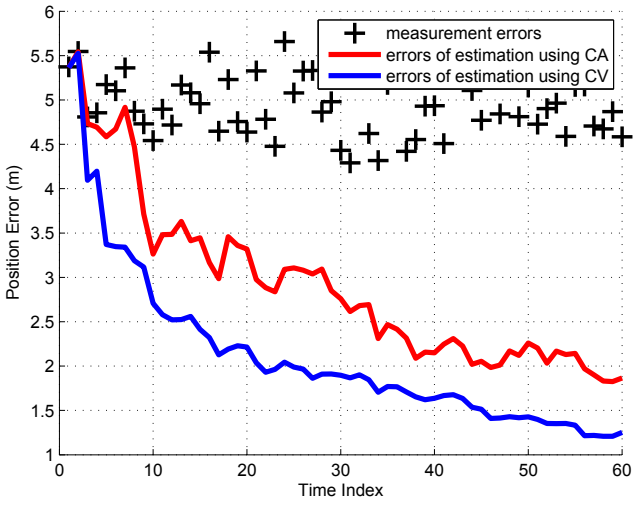

Figure 14: Position estimate and measurement errors for 100 Monte Carlo trials

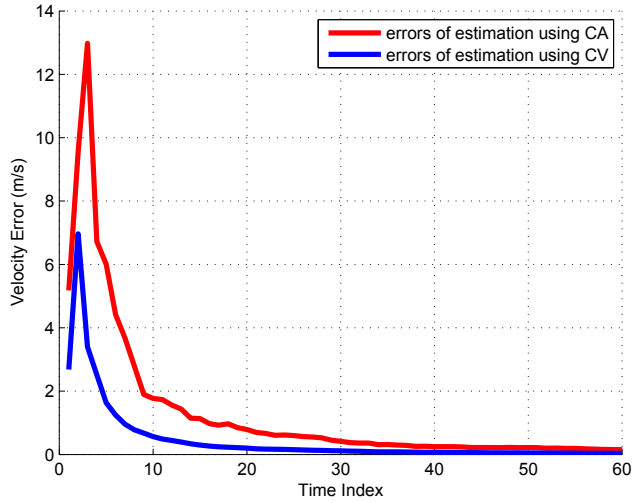

Figure 15: Velocity estimate errors for 100 Monte Carlo trials

Discussion

The test results presented in this subsection tell us another point: for a lower-order movement, we had better utilize a system model matching this movement, not a higher-order system model which seems to be more generalized.

These discussions, together with discussions in the previous subsection, lead to the final point: **establishing a suitable system model is important for achieving desirable estimation performance**. We had better neither use a system model that is not flexible enough to characterize the state evolution (this can be regarded as an **underfitting** problem) nor use a system model that is too generalized and captures useless trends in the state evolution (this can be regarded as an **overfitting** problem).

Arriving at this point, we can conclude this article and leave rest discussions to section 4.

4 Conclusion

In this article i.e. the second part of the planned series "*A Brief Tutorial On Recursive Estimation*", we have explained with concrete examples of vehicle tracking the importance of a suitable system model for an estimation process—*How we can know about the world depends on how we represent the world*; the better can we represent the world, the better can we know about the world.

On the other hand, establishing a suitable system model, it is easy to say than to do. Still consider the examples in this article, how can we know *a priori* the movement pattern of the vehicle? Besides, the vehicle may sometimes be stationary, may sometimes move at constant velocity, and may sometimes move at constant acceleration etc. So using a single system model may not suit various possibilities of vehicle movement patterns. To handle this problem, we may prepare a set of candidate system models and use an on-line identification mechanism to evaluate dynamically the degree of consistency between each candidate system model and the observed vehicle movements; then in the estimation, we can rely more on candidate system models with high degree of consistency while relying less on those with low degree of consistency. This is the basic idea of many multiple-model-based estimation methods, such as the *Interacting Multiple Model* (IMM) [13], which have

already been employed in intelligent vehicle applications [14] [15] [16].

References

[1] H. Li. *A Brief Tutorial On Recursive Estimation With Examples From Intelligent Vehicle Applications (Part I): Basic Spirit And Utilities*. HAL Open Archives, 2014.

[2] H. Li, F. Nashashibi, and G. Toulminet. Localization for intelligent vehicle by fusing mono-camera, low-cost gps and map data. In *IEEE International Conference on Intelligent Transportation Systems*, pages 1657–1662, 2010.

[3] H. Li and F. Nashashibi. Cooperative multi-vehicle localization using split covariance intersection filter. *IEEE Intelligent Transportation Systems Magazine*, 5(2):33–44, 2013.

[4] H. Li and F. Nashashibi. Multi-vehicle cooperative localization using indirect vehicle-to-vehicle relative pose estimation. In *IEEE International Conference on Vehicular Electronics and Safety*, pages 267–272, 2012.

[5] H. Li and F. Nashashibi. Multi-vehicle cooperative perception and augmented reality for driver assistance: A possibility to see through front vehicle. In *IEEE International Conference on Intelligent Transportation Systems*, pages 242–247, 2011.

[6] H. Li, F. Nashashibi, and M. Yang. Split covariance intersection filter: Theory and its application to vehicle localization. *IEEE Transactions on Intelligent Transportation Systems*, 14(4):1860–1871, 2013.

[7] R.E. Kalman. A new approach to linear filtering and prediction problem. *ASME Trans, Ser. D, J. Basic Eng.*, 82:35–45, 1960.

[8] Y. Bar-Shalom and X.R. Li. *Multitarget-multisensor tracking: principles and techniques*. Storrs, CT: University of Connecticut, 1995.

[9] C.C. Wang, C. Thorpe, and S. Thrun. Simultaneous localization, mapping and moving object tracking. *International Journal of Robotics Research*, 26(9):889–916, 2007.

[10] M. Aeberhard, S. Schlichtharle, N. Kaempchen, and T. Bertram. Track-to-track fusion with asynchronous sensors using information matrix fusion for surround environment perception. *IEEE Transactions on Intelligent Transportation Systems*, 13(4):1717–1726, 2012.

[11] M.S. Darms, P.E. Rybski, C. Baker, and C. Urmson. Obstacle detection and tracking for the urban challenge. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):475–485, 2009.

[12] H. Li, F. Nashashibi, B. Lefaudeux, and E. Pollard. Track-to-track fusion using split covariance intersection filter-information matrix filter (scif-imf) for vehicle surrounding environment perception. In *IEEE International Conference on Intelligent Transportation Systems*, pages 1430–1435, 2013.

[13] H.A.P. Blom and Y. Bar-Shalom. The interacting multiple model algorithm for systems with markovian switching coefficients. *IEEE Transactions on Automatic Control*, 33(8):780–783, 1988.

[14] K. Jo, K. Chu, and M. Sunwoo. Interacting multiple model filter-based sensor fusion of gps with in-vehicle sensors for real-time vehicle positioning. *IEEE Transactions on Intelligent Transportation Systems*, 13(1):329–343, 2012.

[15] T.N. Nguyen, B. Michaelis, A. Al-Hamadi, M. Tornow, and M. Meinecke. Stereo-camera-based urban environment perception using occupancy grid and object tracking. *IEEE Transactions on Intelligent Transportation Systems*, 13(1):154–165, 2012.

[16] T.D. Vu. *Vehicle perception: Localization, mapping with detection, classification and tracking of moving objects*. Ph.D. Thesis, Institut National Polytechnique de Grenoble, 2009.