We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900
Open access books available

185,000
International authors and editors

200M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Chapter

Introduction to Kalman Filter and Its Applications

Youngjoo Kim and Hyochoong Bang

Abstract

We provide a tutorial-like description of Kalman filter and extended Kalman filter. This chapter aims for those who need to teach Kalman filters to others, or for those who do not have a strong background in estimation theory. Following a problem definition of state estimation, filtering algorithms will be presented with supporting examples to help readers easily grasp how the Kalman filters work. Implementations on INS/GNSS navigation, target tracking, and terrain-referenced navigation (TRN) are given. In each example, we discuss how to choose, implement, tune, and modify the algorithms for real world practices. Source codes for implementing the examples are also provided. In conclusion, this chapter will become a prerequisite for other contents in the book.

Keywords: Kalman filter, extended Kalman filter, INS/GNSS navigation, target tracking, terrain-referenced navigation

1. Introduction

Kalman filtering is an algorithm that provides estimates of some unknown variables given the measurements observed over time. Kalman filters have been demonstrating its usefulness in various applications. Kalman filters have relatively simple form and require small computational power. However, it is still not easy for people who are not familiar with estimation theory to understand and implement the Kalman filters. Whereas there exist some excellent literatures such as [1] addressing derivation and theory behind the Kalman filter, this chapter focuses on a more practical perspective.

Following two chapters will devote to introduce algorithms of Kalman filter and extended Kalman filter, respectively, including their applications. With linear models with additive Gaussian noises, the Kalman filter provides optimal estimates. Navigation with a global navigation satellite system (GNSS) will be provided as an implementation example of the Kalman filter. The extended Kalman filter is utilized for nonlinear problems like bearing-angle target tracking and terrain-referenced navigation (TRN). How to implement the filtering algorithms for such applications will be presented in detail.

IntechOpen

2. Kalman filter

2.1 Problem definition

Kalman filters are used to estimate states based on linear dynamical systems in state space format. The process model defines the evolution of the state from time $k-1$ to time k as:

$$\boldsymbol{x}_k = F\boldsymbol{x}_{k-1} + B\boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1} \tag{1}$$

where F is the state transition matrix applied to the previous state vector $\boldsymbol{x}_{k-1}$, B is the control-input matrix applied to the control vector $\boldsymbol{u}_{k-1}$, and $\boldsymbol{w}_{k-1}$ is the process noise vector that is assumed to be zero-mean Gaussian with the covariance Q, i.e., $\boldsymbol{w}_{k-1} \sim \mathcal{N}(0, Q)$.

The process model is paired with the measurement model that describes the relationship between the state and the measurement at the current time step k as:

$$\mathbf{z}_k = H\boldsymbol{x}_k + \boldsymbol{\nu}_k \tag{2}$$

where $\mathbf{z}_k$ is the measurement vector, H is the measurement matrix, and $\boldsymbol{\nu}_k$ is the measurement noise vector that is assumed to be zero-mean Gaussian with the covariance R, i.e., $\boldsymbol{\nu}_k \sim \mathcal{N}(0, R)$. Note that sometimes the term "measurement" is called "observation" in different literature.

The role of the Kalman filter is to provide estimate of $\boldsymbol{x}_k$ at time k, given the initial estimate of $\boldsymbol{x}_0$, the series of measurement, $\mathbf{z}_1$, $\mathbf{z}_2$, ..., $\mathbf{z}_k$, and the information of the system described by F, B, H, Q, and R. Note that subscripts to these matrices are omitted here by assuming that they are invariant over time as in most applications. Although the covariance matrices are supposed to reflect the statistics of the noises, the true statistics of the noises is not known or not Gaussian in many practical applications. Therefore, Q and R are usually used as tuning parameters that the user can adjust to get desired performance.

2.2 Kalman filter algorithm

Kalman filter algorithm consists of two stages: prediction and update. Note that the terms "prediction" and "update" are often called "propagation" and "correction," respectively, in different literature. The Kalman filter algorithm is summarized as follows:

Prediction:

Predicted state estimate	$\hat{\boldsymbol{x}}_k^- = F\hat{\boldsymbol{x}}_{k-1}^+ + B\boldsymbol{u}_{k-1}$
Predicted error covariance	$P_k^- = FP_{k-1}^+F^T + Q$

Update:

Measurement residual	$\tilde{\boldsymbol{y}}_k = \mathbf{z}_k - H\hat{\boldsymbol{x}}_k^-$
Kalman gain	$K_k = P_k^-H^T\left(R + HP_k^-H^T\right)^{-1}$
Updated state estimate	$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + K_k\tilde{\boldsymbol{y}}$
Updated error covariance	$P_k^+ = (I - K_kH)P_k^-$

In the above equations, the hat operator, ˆ, means an estimate of a variable. That is, $\hat{\boldsymbol{x}}$ is an estimate of $\boldsymbol{x}$. The superscripts – and + denote predicted (prior) and updated (posterior) estimates, respectively.

The predicted state estimate is evolved from the updated previous updated state estimate. The new term P is called state error covariance. It encrypts the error covariance that the filter thinks the estimate error has. Note that the covariance of a random variable $\boldsymbol{x}$ is defined as $\text{cov}(\boldsymbol{x}) = \mathbb{E}\left[(\boldsymbol{x} - \hat{\boldsymbol{x}})(\boldsymbol{x} - \hat{\boldsymbol{x}})^T\right]^T$ where $\mathbb{E}$ denotes the expected (mean) value of its argument. One can observe that the error covariance becomes larger at the prediction stage due to the summation with Q, which means the filter is more uncertain of the state estimate after the prediction step.

In the update stage, the measurement residual $\widetilde{\boldsymbol{y}}_k$ is computed first. The measurement residual, also known as innovation, is the difference between the true measurement, $\mathbf{z}_k$, and the estimated measurement, $H\hat{\boldsymbol{x}}_k^-$. The filter estimates the current measurement by multiplying the predicted state by the measurement matrix. The residual, $\widetilde{\boldsymbol{y}}_k$, is later then multiplied by the Kalman gain, K_k, to provide the correction, $K_k\widetilde{\boldsymbol{y}}_{\text{k}}$, to the predicted estimate $\hat{\boldsymbol{x}}_k^-$. After it obtains the updated state estimate, the Kalman filter calculates the updated error covariance, P_k^+, which will be used in the next time step. Note that the updated error covariance is smaller than the predicted error covariance, which means the filter is more certain of the state estimate after the measurement is utilized in the update stage.

We need an initialization stage to implement the Kalman filter. As initial values, we need the initial guess of state estimate, $\hat{\boldsymbol{x}}_0^+$, and the initial guess of the error covariance matrix, P_0^+. Together with Q and R, $\hat{\boldsymbol{x}}_0^+$ and P_0^+ play an important role to obtain desired performance. There is a rule of thumb called "initial ignorance," which means that the user should choose a large P_0^+ for quicker convergence. Finally, one can obtain implement a Kalman filter by implementing the prediction and update stages for each time step, $k = 1, 2, 3, \ldots$, after the initialization of estimates.

Note that Kalman filters are derived based on the assumption that the process and measurement models are linear, i.e., they can be expressed with the matrices F, B, and H, and the process and measurement noise are additive Gaussian. Hence, a Kalman filter provides optimal estimate only if the assumptions are satisfied.

2.3 Example

An example for implementing the Kalman filter is navigation where the vehicle state, position, and velocity are estimated by using sensor output from an inertial measurement unit (IMU) and a global navigation satellite system (GNSS) receiver. In this example, we consider only position and velocity, omitting attitude information. The three-dimensional position and velocity comprise the state vector:

$$\boldsymbol{x} = \left[\boldsymbol{p}^T, \boldsymbol{v}^T\right]^T \tag{3}$$

where $\boldsymbol{p} = \left[p_x, p_y, p_{\mathbf{z}}\right]^T$ is the position vector and $\boldsymbol{v} = \left[v_x, v_y, v_{\mathbf{z}}\right]^T$ is the velocity vector whose elements are defined in x, y, z axes. The state in time k can be predicted by the previous state in time $k-1$ as:

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{p}_k \\ \boldsymbol{v}_k \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_{k-1} + \boldsymbol{v}_{k-1}\Delta t + \frac{1}{2}\widetilde{\boldsymbol{a}}_{k-1}\Delta t^2 \\ \boldsymbol{v}_{k-1} + \widetilde{\boldsymbol{a}}_{k-1}\Delta t \end{bmatrix} \tag{4}$$

where $\widetilde{\boldsymbol{a}}_{k-1}$ is the acceleration applied to the vehicle. The above equation can be rearranged as:

$$\boldsymbol{x}_k = \begin{bmatrix} I_{3\times3} & I_{3\times3}\Delta t \\ 0_{3\times3} & I_{3\times3} \end{bmatrix} \boldsymbol{x}_{k-1} + \begin{bmatrix} \frac{1}{2} I_{3\times3}\Delta t^2 \\ I_{3\times3}\Delta t \end{bmatrix} \widetilde{\boldsymbol{a}}_{k-1} \tag{5}$$

where $I_{3\times3}$ and $0_{3\times3}$ denote 3×3 identity and zero matrices, respectively. The process noise comes from the accelerometer output, $\boldsymbol{a}_{k-1} = \widetilde{\boldsymbol{a}}_{k-1} + \boldsymbol{e}_{k-1}$, where $\boldsymbol{e}_{k-1}$ denotes the noise of the accelerometer output. Suppose $\boldsymbol{e}_{k-1} \sim \mathcal{N}\left(0, I_{3\times3}\sigma_e^2\right)$. From the covariance relationship, $\mathrm{Cov}(A\boldsymbol{x}) = A\Sigma A^T$ where $\mathrm{Cov}(\boldsymbol{x}) = \Sigma$, we get the covariance matrix of the process noise as:

$$Q = \begin{bmatrix} \frac{1}{2} I_{3\times3}\Delta t^2 \\ I_{3\times3}\Delta t \end{bmatrix} I_{3\times3}\sigma_e^2 \begin{bmatrix} \frac{1}{2} I_{3\times3}\Delta t^2 \\ I_{3\times3}\Delta t \end{bmatrix}^T = \begin{bmatrix} \frac{1}{4} I_{3\times3}\Delta t^4 & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3}\Delta t^2 \end{bmatrix} \sigma_e^2 \tag{6}$$

Now, we have the process model as:

$$\boldsymbol{x}_k = F\boldsymbol{x}_{k-1} + B\boldsymbol{a}_{k-1} + \boldsymbol{w}_{k-1} \tag{7}$$

where

$$F = \begin{bmatrix} I_{3\times3} & I_{3\times3}\Delta t \\ 0_{3\times3} & I_{3\times3} \end{bmatrix} \tag{8}$$

$$B = \begin{bmatrix} \frac{1}{2} I_{3\times3}\Delta t^2 \\ I_{3\times3}\Delta t \end{bmatrix} \tag{9}$$

$$\boldsymbol{w}_{k-1} \sim \mathcal{N}(0, Q) \tag{10}$$

The GNSS receiver provides position and velocity measurements corrupted by measurement noise $\boldsymbol{\nu}_k$ as:

$$\mathbf{z}_k = \begin{bmatrix} \boldsymbol{p}_k \\ \boldsymbol{v}_k \end{bmatrix} + \boldsymbol{\nu}_k \tag{11}$$

It is straightforward to derive the measurement model as:

$$\mathbf{z}_k = H\boldsymbol{x}_k + \boldsymbol{\nu}_k \tag{12}$$

where

$$H = I_{6\times6} \tag{13}$$

$$\boldsymbol{\nu}_k \sim \mathcal{N}(0, R) \tag{14}$$

In order to conduct a simulation to see how it works, let us consider $N = 20$ time steps ($k = 1, 2, 3, \ldots, N$) with $\Delta t = 1$. It is recommended to generate a time history of true state, or a true trajectory, first. The most convenient way is to generate the series of true accelerations over time and integrate them to get true velocity and position. In this example, the true acceleration is set to zero and the vehicle is moving with a constant velocity, $\boldsymbol{v}_k = [5, 5, 0]^T$ for all $k = 1, 2, 3, \ldots, N$, from the

initial position, $\boldsymbol{p}_0 = [0, 0, 0]$. Note that one who uses the Kalman filter to estimate the vehicle state is usually not aware whether the vehicle has a constant velocity or not. This case is not different from nonzero acceleration case in perspective of this Kalman filter models. If the filter designer (you) has some prior knowledge of the vehicle maneuver, process models can be designed in different forms for best describing various maneuvers as in [2].

We need to generate noise of acceleration output and GNSS measurements for every time step. Suppose the acceleration output, GNSS position, and GNSS velocity are corrupted with noise with variances of 0.3^2, 3^2, and 0.03^2, respectively. For each axis, one can use MATLAB function randn or normrnd for generating the Gaussian noise.

The process noise covariance matrix, Q, and measurement noise covariance matrix, R, can be constructed following the real noise statistics described above to get the best performance. However, have in mind that in real applications, we do not know the real statistics of the noises and the noises are often not Gaussian. Common practice is to conservatively set Q and R slightly larger than the expected values to get robustness.

Let us start filtering with the initial guesses

$$\hat{\boldsymbol{x}}_0^+ = [2, -2, 0, 5, 5.1, 0.1]^T \tag{15}$$

$$P_0^+ = \begin{bmatrix} I_{3\times3}4^2 & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3}0.4^2 \end{bmatrix} \tag{16}$$

and noise covariance matrices

$$Q = \begin{bmatrix} \frac{1}{4}I_{3\times3}\Delta t^4 & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3}\Delta t^2 \end{bmatrix} 0.3^2 \tag{17}$$

$$R = \begin{bmatrix} I_{3\times3}3^2 & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3}0.03^2 \end{bmatrix} \tag{18}$$

where Q and R are constant for every time step. The more uncertain your initial guess for the state is, the larger the initial error covariance should be.

In this simulation, $M = 100$ Monte-Carlo runs were conducted. A single run is not sufficient for verifying the statistic characteristic of the filtering result because each sample of a noise differs whenever the noise is sampled from a given distribution, and therefore, every simulation run results in different state estimate. The repetitive Monte-Carlo runs enable us to test a number of different noise samples for each time step.

The time history of estimation errors of two Monte-Carlo runs is depicted in **Figure 1**. We observe that the estimation results of different simulation runs are different even if the initial guess for the state estimate is the same. You can also run the Monte-Carlo simulation with different initial guesses (sampled from a distribution) for the state estimate.

The standard deviation of the estimation errors and the estimated standard deviation for x-axis position and velocity are drawn in **Figure 2**. The standard deviation of the estimation error, or the root mean square error (RMSE), can be obtained by computing standard deviation of M estimation errors for each time step. The estimated standard deviation was obtained by taking squared root of the

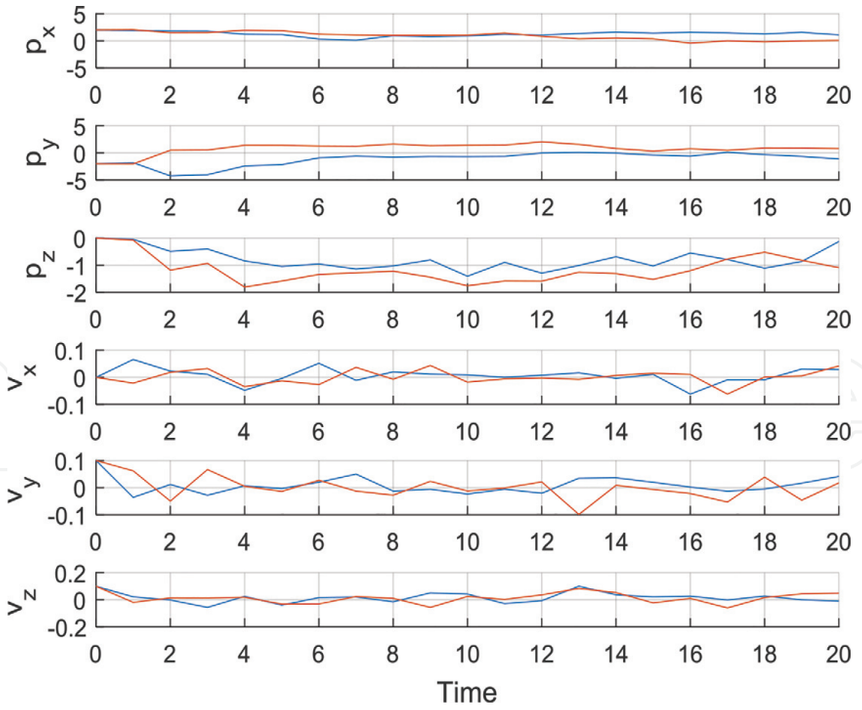


Figure 1.
Time history of estimation errors.

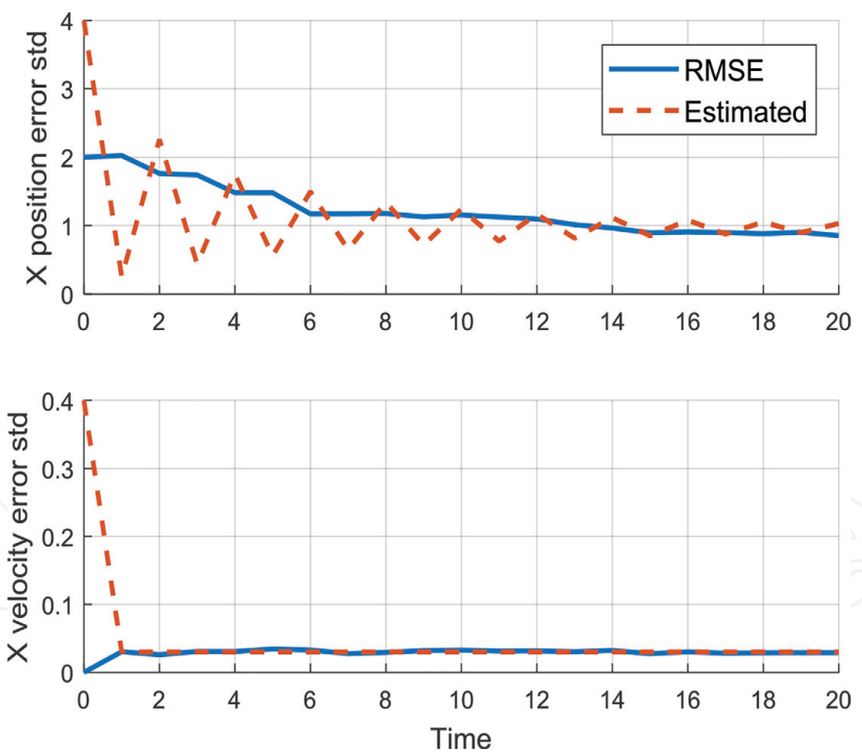


Figure 2.
Actual and estimated standard deviation for x-axis estimate errors.

corresponding diagonal term of P_k^+. Drawing the estimated standard deviation for each axis is possible because the state estimates are independent to each other in this example. A care is needed if P_k^+ has nonzero off-diagonal terms. The estimated standard deviation and the actual standard deviation of estimate errors are very similar. In this case, the filter is called consistent. Note that the estimated error covariance matrix is affected solely by P_0^+, Q, and R, judging from the Kalman filter

algorithm. Different settings to these matrices will result in different P_k^+ and therefore different state estimates.

In real applications, you will be able to acquire only the estimated covariance because you will hardly have a chance to conduct Monte-Carlo runs. Also, getting a good estimate of Q and R is often difficult. One practical approach to estimate the noise covariance matirces is the autocovariance least-squares (ALS) technique [3] or an adaptive Kalman filter where the noise covariance matrices are adjusted in real time can be used [4].

Source code of MATLAB implementation for this example can be found in [5]. It is recommended for the readers to change the parameters and aircraft trajectory by yourself and see what happens.

3. Extended Kalman filter

3.1 Problem definition

Suppose you have a nonlinear dynamic system where you are not able to define either the process model or measurement model with multiplication of vectors and matrices as in (1) and (2). The extended Kalman filter provides us a tool for dealing with such nonlinear models in an efficient way. Since it is computationally cheaper than other nonlinear filtering methods such as point-mass filters and particle filters, the extended Kalman filter has been used in various real-time applications like navigation systems.

The extended Kalman filter can be viewed as a nonlinear version of the Kalman filter that linearized the models about a current estimate. Suppose we have the following models for state transition and measurement

$$\boldsymbol{x}_k = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) + \boldsymbol{w}_{k-1} \tag{19}$$

$$\mathbf{z}_k = \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{\nu}_k \tag{20}$$

where $\boldsymbol{f}$ is the function of the previous state, $\boldsymbol{x}_{k-1}$, and the control input, $\boldsymbol{u}_{k-1}$, that provides the current state $\boldsymbol{x}_k$. $\boldsymbol{h}$ is the measurement function that relates the current state, $\boldsymbol{x}_k$, to the measurement $\mathbf{z}_k$. $\boldsymbol{w}_{k-1}$ and $\boldsymbol{\nu}_k$ are Gaussian noises for the process model and the measurement model with covariance Q and R, respectively.

3.2. Extended Kalman filter algorithm

All you need is to obtain the Jacobian matrix, first-order partial derivative of a vector function with respect to a vector, of each model in each time step as:

$$F_{k-1} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_{k-1}^+, \boldsymbol{u}_{k-1}} \tag{21}$$

$$H_k = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_k^-} \tag{22}$$

Note the subscripts of F and H are maintained here since the matrices are often varying with different values of the state vector for each time step. By doing this, you linearize the models about the current estimate. The filter algorithm is very similar to Kalman filter.

Prediction:

Predicted state estimate	$\hat{\boldsymbol{x}}_k^- = \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}^+, \boldsymbol{u}_{k-1})$
Predicted error covariance	$P_k^- = F_{k-1}P_{k-1}^+F_{k-1}^T + Q$

Update:

Measurement residual	$\tilde{\boldsymbol{y}}_k = \mathbf{z}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_k^-)$
Kalman gain	$K_k = P_k^- H_k^T \left(R + H_k P_k^- H_k^T\right)^{-1}$
Updated state estimate	$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + K_k\tilde{\boldsymbol{y}}$
Updated error covariance	$P_k^+ = (I - K_k H_k)P_k^-$

As in the Kalman filter algorithm, the hat operator, ˆ, means an estimate of a variable. That is, $\hat{\boldsymbol{x}}$ is an estimate of $\boldsymbol{x}$. The superscripts – and + denote predicted (prior) and updated (posterior) estimates, respectively. The main difference from the Kalman filter is that the extended Kalman filter obtains predicted state estimate and predicted measurement by the nonlinear functions $\boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1})$ and $\boldsymbol{h}(\boldsymbol{x}_k)$, respectively.

3.3 Example

3.3.1 Target tracking

We are going to estimate a 3-dimensional target state (position and velocity) by using measurements provided by a range sensor and an angle sensor. For example, a radar system can provide range and angle measurement and a combination of a camera and a rangefinder can do the same. We define the target state as:

$$\boldsymbol{x} = \left[\boldsymbol{p}^T, \boldsymbol{v}^T\right]^T \tag{23}$$

where $\boldsymbol{p}$ and $\boldsymbol{v}$ denote position and velocity of the target, respectively. The system model is described as a near-constant-velocity model [2] in discrete time space by:

$$\boldsymbol{x}_k = \begin{bmatrix} \boldsymbol{p}_k \\ \boldsymbol{v}_k \end{bmatrix} = \boldsymbol{f}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}) = \begin{bmatrix} \boldsymbol{p}_{k-1} + \boldsymbol{v}_{k-1}\Delta t \\ \boldsymbol{v}_{k-1} \end{bmatrix} + \boldsymbol{w}_{k-1} \tag{24}$$

The process noise has the covariance of $\boldsymbol{w}_{k-1} \sim \mathcal{N}(0, Q)$ where

$$Q = \begin{bmatrix} 0_{3\times3} & & 0_{3\times3} & \\ & \sigma_x^2 & 0 & 0 \\ 0_{3\times3} & 0 & \sigma_y^2 & 0 \\ & 0 & 0 & \sigma_{\mathbf{z}}^2 \end{bmatrix} \tag{25}$$

and σ_x, σ_y, and $\sigma_{\mathbf{z}}$ are the standard deviations of the process noise on the velocity in x, y, and z directions, respectively.

The measurement vector is composed of line-of-sight angles to the target, $\mathcal{A}$ and $\mathcal{E}$, and the range, $\mathcal{R}$, to the target. The relationship between the measurement and the relative target state with respect to the sensor comprises the measurement model as:

$$\mathbf{z}_k = \begin{bmatrix} \mathcal{A} \\ \mathcal{E} \\ \mathcal{R} \end{bmatrix} = \boldsymbol{h}(\boldsymbol{x}_k) = \begin{bmatrix} atan\left(\dfrac{x_t - x_s}{y_t - y_s}\right) \\ atan\left(\dfrac{\mathbf{z}_t - \mathbf{z}_s}{\sqrt{(x_t - x_s)^2 + (y_t - y_s)^2}}\right) \\ \sqrt{(x_t - x_s)^2 + (y_t - y_s)^2 + (\mathbf{z}_t - \mathbf{z}_s)^2} \end{bmatrix} + \boldsymbol{\nu}_k \tag{26}$$

where $p_k = [x_t, y_t, \mathbf{z}_t]^T$ is the position vector of the target and $[x_s, y_s, \mathbf{z}_s]^T$ is the position vector of the sensor. The target position is the variable in this measurement model. Note that the measurement has nonlinear relationship with the target state. This cannot be expressed in a matrix form as in (2) whereas the process model can be. If at least one model is nonlinear, we should use nonlinear filtering technique. In order to apply extended Kalman filter to this problem, let us take first derivatives of the process model and measurement model as:

$$F_{k-1} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}^+_{k-1}, \boldsymbol{u}_{k-1}} = \begin{bmatrix} I_{3\times3} & I_{3\times3}\Delta t \\ 0_{3\times3} & I_{3\times3} \end{bmatrix} \tag{27}$$

$$H_k = \left.\frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}^-_k} = \begin{bmatrix} \dfrac{y}{x^2+y^2} & \dfrac{-x}{x^2+y^2} & 0 & \\ \dfrac{-xz}{(x^2+y^2+\mathbf{z}^2)\sqrt{x^2+y^2}} & \dfrac{-yz}{(x^2+y^2+\mathbf{z}^2)\sqrt{x^2+y^2}} & \dfrac{1}{\sqrt{x^2+y^2}} & 0_{3\times3} \\ \dfrac{x}{\sqrt{x^2+y^2+\mathbf{z}^2}} & \dfrac{y}{\sqrt{x^2+y^2+\mathbf{z}^2}} & \dfrac{\mathbf{z}}{\sqrt{x^2+y^2+\mathbf{z}^2}} & \end{bmatrix} \tag{28}$$

where $[x, y, \mathbf{z}]^T = [x_t - x_s, y_t - y_s, \mathbf{z}_t - \mathbf{z}_s]^T$ is the relative position vector. Note that the matrix H_k varies with different values of $[x, y, \mathbf{z}]^T$ on which the filtering result will, therefore, depend. Thus, one can plan the trajectory of the sensor to get a better filtering result [6]. Developing such a method is one of active research topics.

In the simulation, the sensor is initially located at $[x_s, y_s, \mathbf{z}_s]^T = [40, 20, 50]^T$ and the sensor is moving in a circular pattern with a radius of 20 centered at $[20, 20, 50]^T$. The initial state of the target is $\boldsymbol{x}_0 = [10, -10, 0, -1, -2, 0]^T$. The sensor is moving with a constant velocity of $[-1, -2, 0]^T$. The trajectory of the target and the sensor is shown in **Figure 3**. Note that this is the case where we are aware that the target has a constant velocity, unlike the example in Section 2.3, which is why we modeled the state transition as the near-constant-velocity model in (4). Let us consider $N = 20$ time steps ($k = 1, 2, 3, ..., N$) with $\Delta t = 1$. Suppose the measurements are corrupted with a Gaussian noise whose standard deviation is $[0.02, 0.02, 1.0]^T$.

In the filter side, the covariance matrix for the process noise can be set as:

$$Q = \begin{bmatrix} 0_{3\times3} & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3}\sigma_v^2 \end{bmatrix} \tag{29}$$

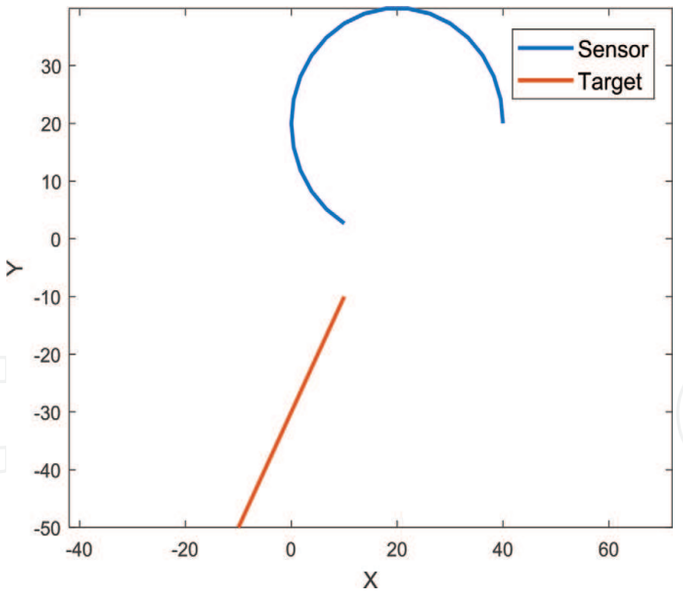


Figure 3.
Trajectory of the sensor and the target.

where $\sigma_v = 5$ is the tuning parameter that denotes how uncertain the velocity estimate is. The measurement covariance matrix was constructed following the real noise statistics as:

$$R = \begin{bmatrix} 0.02^2 & 0 & 0 \\ 0 & 0.02^2 & 0 \\ 0 & 0 & 1.0^2 \end{bmatrix} \tag{30}$$

$M = 100$ Monte-Carlo runs were conducted with the following initial guesses:

$$\hat{\boldsymbol{x}}_0^+ = \boldsymbol{x}_0 + \text{normrnd}(0, [1, 1, 0, 0, 0, 0]) \tag{31}$$

$$P_0^+ = \begin{bmatrix} I_{3\times3}1^2 & 0_{3\times3} \\ 0_{3\times3} & I_{3\times3}0.1^2 \end{bmatrix} \tag{32}$$

The above equation means that the error of the initial guess for the target state is randomly sampled from a Gaussian distribution with a standard deviation of $[1, 1, 0, 0, 0, 0]$.

Time history of an estimation result for x-axis position and velocity is drawn together with the true value in **Figure 4**. The shape of the line will be different at each run. The statistical result can be shown as **Figure 5**. Note that the filter worked inconsistently with the estimated error covariance different from the actual value. This is because the process error covariance is set to a very large number. In this example, the large process error covariance is the only choice a user can make because the measurement cannot correct the velocity. One can notice that the measurement Eq. (26) has no term dependent on the velocity, and therefore, matrix H in (28) has zero elements on the right side of the matrix where the derivatives of the measurement equation with respect to velocity are located. As a result, the measurement residual has no effect on velocity correction. In this case, we say the system has no observability on velocity. In practice, this problem can be mitigated by setting the process noise covariance to a large number so that the filter believes the measurement is more reliable. In this way, we can prevent at least the position estimate from diverging.

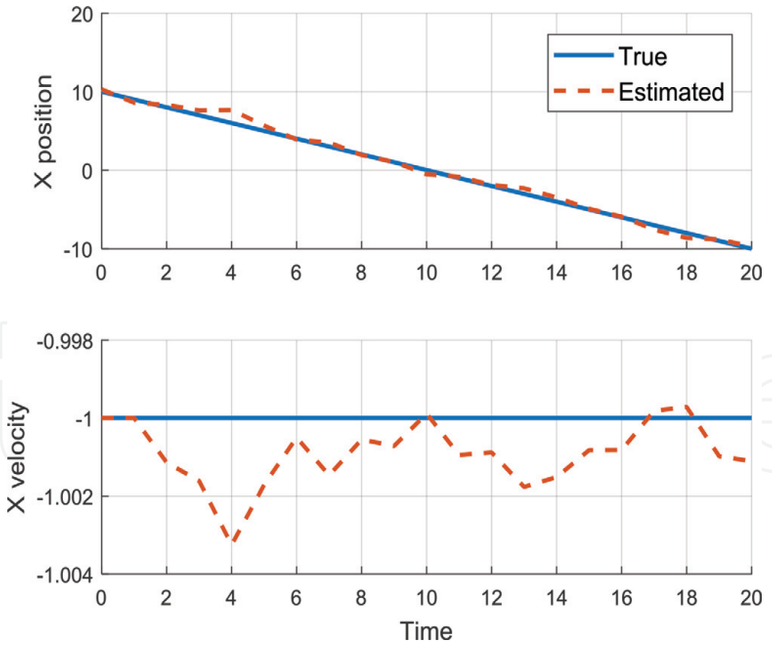


Figure 4.
Time history of an estimation result for x-axis position and velocity.

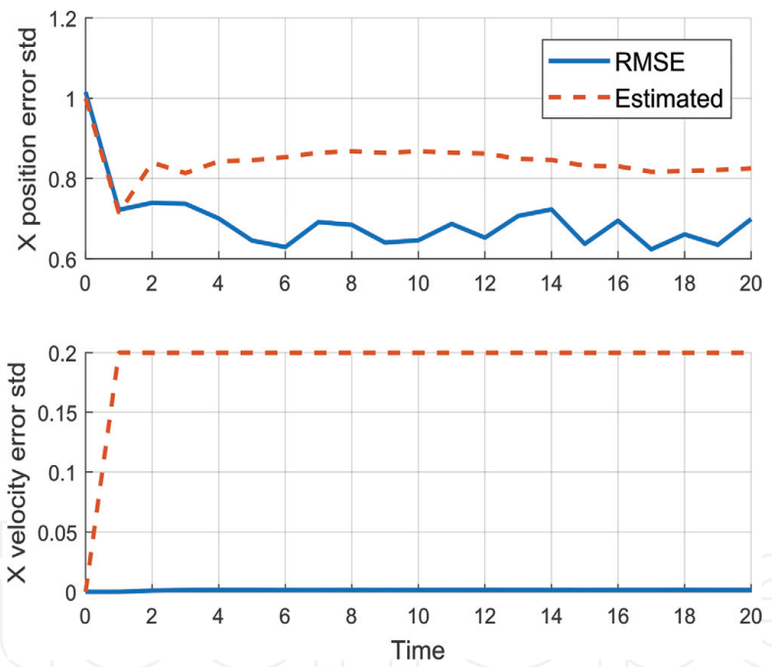


Figure 5.
Actual and estimated standard deviation for x axis estimate errors.

Source code of MATLAB implementation for this example can be found in [5]. It is recommended for the readers to change the parameters and trajectories by yourself and see what happens.

3.3.2 Terrain-referenced navigation

Terrain-referenced navigation (TRN), also known as terrain-aided navigation (TAN), provides positioning data by comparing terrain measurements with a digital elevation model (DEM) stored on an on-board computer of an aircraft. The TRN algorithm blends a navigational solution from an inertial navigation system (INS)

with the measured terrain profile underneath the aircraft. Terrain measurements have generally been obtained by using radar altimeters. TRN systems using cameras [7], airborne laser sensors [8], and interferometric radar altimeters [9] have also been addressed. Unlike GNSS's, TRN systems are resistant to electronic jamming and interference, and are able to operate in a wide range of weather conditions. Thus, TRN systems are expected to be alternative/supplement systems to GNSS's.

The movement of the aircraft is modeled by the following Markov process:

$$\boldsymbol{x}_k = \boldsymbol{x}_{k-1} + \boldsymbol{u}_{k-1} + \boldsymbol{w}_{k-1} \tag{33}$$

where $\boldsymbol{x}_{k-1}$, $\boldsymbol{u}_{k-1}$, and $\boldsymbol{w}_{k-1}$ denote the state vector, the relative movement, and the additive Gaussian process noise, respectively, at time $k-1$. $\boldsymbol{x}_k = [\phi, \lambda]^T$ is a two-dimensional state vector, which denotes the aircraft's horizontal position. Estimates of the relative movement (velocity) are provided by the INS and their error is absorbed into $\boldsymbol{w}_{k-1}$ to limit the dimensionality of the state. The simple model in (33) is considered realistic without details of INS integration if an independent attitude solution is available so that the velocity can be resolved in an earth-fixed frame [10]. The estimation models we deal with belong to the TRN filter block in **Figure 6**, taking relative movement information from the INS as $\boldsymbol{u}_k$.

Typical TRN systems utilize measurements of the terrain elevation underneath an aircraft. The terrain elevation measurement is modeled as:

$$\mathbf{z}_k = h(\boldsymbol{x}_k) + \upsilon_k \tag{34}$$

where $h(\boldsymbol{x}_k)$ denotes terrain elevation from the DEM evaluated at the horizontal position, $\boldsymbol{x}_k$, and υ_k denotes the additive Gaussian measurement noise. The elevation measurement is obtained by subtracting the ground clearance measurement from a radar altimeter, h_r, from the barometric altimeter measurement, h_b. υ_k contains errors of the radar altimeter, barometric altimeter, and DEM. The ground clearance and the barometric altitude correspond to the above ground level (AGL) height and the mean sea level (MSL) height, respectively. The relationship between the measurements is depicted in **Figure 7**. Note that the terrain elevation that comprises the measurement model in (34) is highly nonlinear.

The process model in (33) and the measurement model in (34) can be linearized as:

$$F_{k-1} = \left.\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}^+_{k-1}, \boldsymbol{u}_{k-1}} = I_{2\times 2} \tag{35}$$

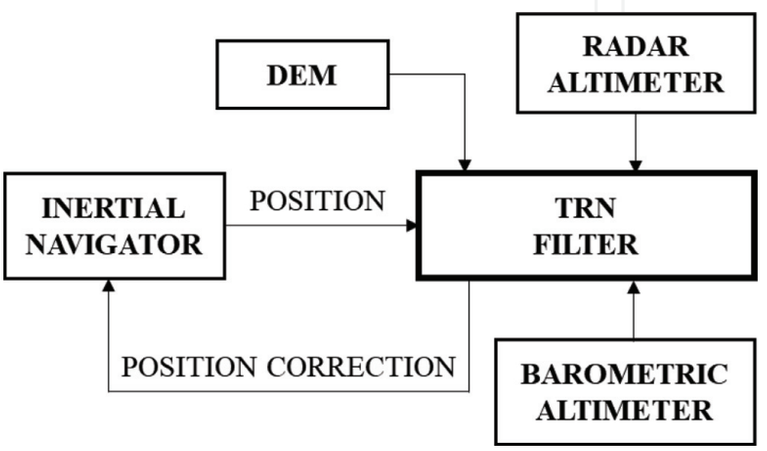


Figure 6.
Conventional TRN structure.

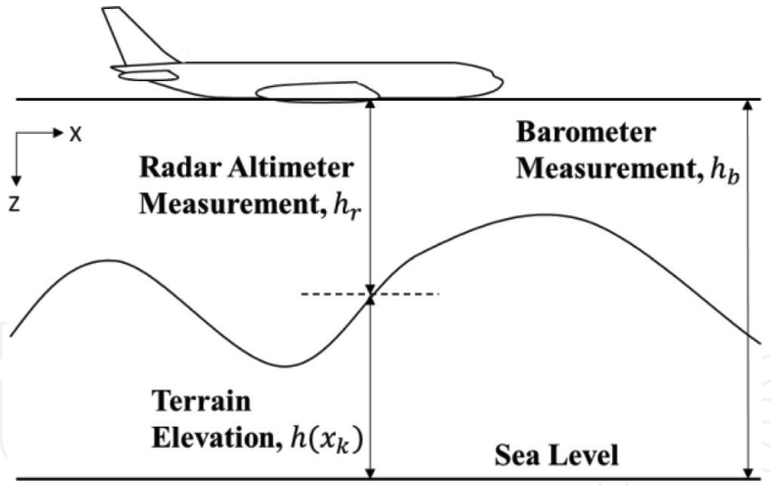


Figure 7.
Relationship between measurements in TRN.

$$H_k = \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}_k^-} = \begin{bmatrix} \frac{\partial D(\phi,\lambda)}{\partial \phi} & \frac{\partial D(\phi,\lambda)}{\partial \lambda} \end{bmatrix} \tag{36}$$

where $D(\phi, \lambda)$ denotes the terrain elevation from the DEM on the horizontal position $[\phi, \lambda]^T$.

The DEMs are essentially provided as matrices containing grid-spaced elevation data. For obtaining finer-resolution data, interpolation techniques are often used to estimate the unknown value in between the grid points. One of the simplest methods is linear interpolation. Linear interpolation is quick and easy, but it is not very precise. A generalization of linear interpolation is polynomial interpolation. Polynomial interpolation expresses data points as higher degree polynomial. Polynomial interpolation overcomes most of the problems of linear interpolation. However, calculating the interpolating polynomial is computationally expensive. Furthermore, the shape of the resulting curve may be different to what is known about the data, especially for very high or low values of the independent variable. These disadvantages can be resolved by using spline interpolation. Spline interpolation uses low-degree polynomials in each of the data intervals and let the polynomial pieces fit smoothly together. That is, its second derivative is zero at the grid points (see [11] for more details). Classical approach to use polynomials of degree 3 is called cubic spline. Because the elevation data are contained in a two-dimensional array, bilinear or bicubic interpolation are generally used. Interpolation for two-dimensional gridded data can be realized by interp2 function in MATLAB. Cubic spline interpolation is used in this example.

The DEM we are using in this example has a 100×100 grid with a resolution of 30. The profile of the DEM can be depicted as **Figure 8**. The figure represents contours of the terrain where brighter color denotes regions with higher altitude. The point $(20, 10)$ in the grid corresponds to the position $[600, 300]^T$ in the navigation frame.

An aircraft, initially located at $\boldsymbol{x}_0 = [400, 400]^T$, is moving by 20 every time step in x direction. The aircraft is equipped with a radar altimter and a barometric altimter, which are used for obtaining the terrain elevation. This measured terrain elevation is compared to the DEM to estimate the vehicle's position.

The process noise $\boldsymbol{w}_{k-1}$ is a zero-mean Gaussian noise with the standard deviation of $[0.5, 0.5]^T$. The radar altimeter is corrupted with a zero-mean Gaussian noise

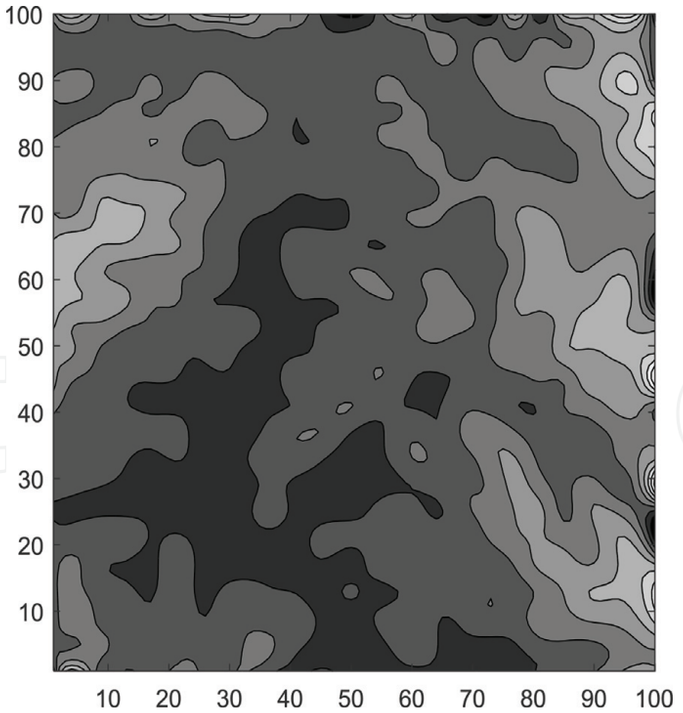


Figure 8.
Contour representation of terrain profile.

with the standard deviation of 3. The matrices Q and R are following the real statistics of the noises as:

$$Q = \begin{bmatrix} 0.5^2 & 0 \\ 0 & 0.5^2 \end{bmatrix} \tag{37}$$

$$R = 3^2 \tag{38}$$

Let us consider $N = 100$ time steps ($k = 1, 2, 3, ..., N$) with $\Delta t = 1$. $M = 100$ Monte-Carlo runs were conducted with the following initial guesses:

$$\hat{\boldsymbol{x}}_0^+ = \boldsymbol{x}_0 + \text{normrnd}(0, [50, 50]) \tag{39}$$

$$P_0^+ = \begin{bmatrix} 50^2 & 0 \\ 0 & 50^2 \end{bmatrix} \tag{40}$$

The above equation means the error of the initial guess for the target state is randomly sampled from a Gaussian distribution with a standard deviation of $[50, 50]$.

The time history of RMSE of the navigation is shown in **Figure 9**. One can observe the RMSE converges relatively slower than other examples. Because the TRN estimates 2D position by using the height measurements, it often lacks information on the vehicle state. Moreover, note that the extended Kalman filter linearizes the terrain model and deals with the slope that is effective locally. If the gradient of the terrain is zero, the measurement matrix H has zero-diagonal terms that has zero effect on the state correction. In this case, the measurement is called ambiguous [12] and this ambiguous measurement often causes filter degradation and divergence even in nonlinear filtering techniques. With highly nonlinear terrain models, TRN systems have recently been constructed with other nonlinear filtering methods such as point-mass filters and particle filters, rather than extended Kalman filters.

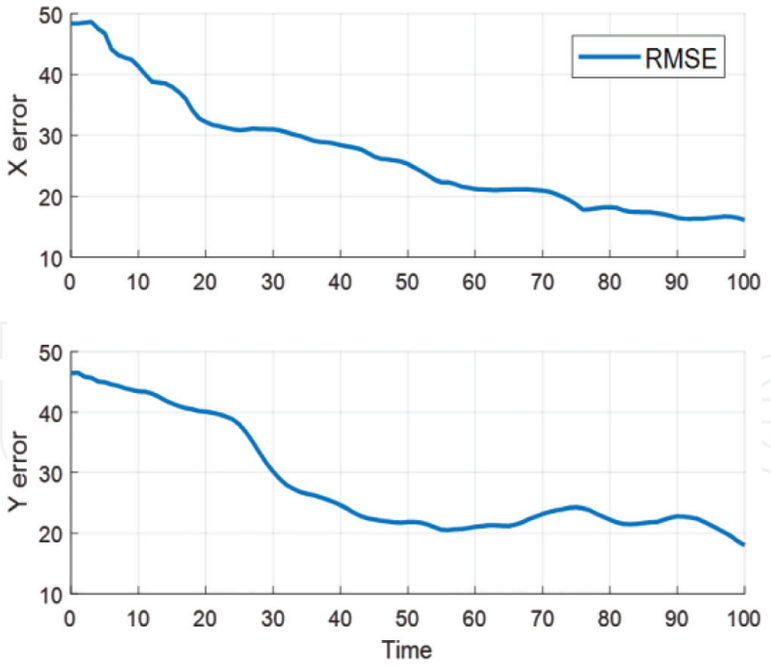


Figure 9.
Time history of RMSE.

Source code of MATLAB implementation for this example can be found in [5]. It is recommended for the readers to change the parameters and aircraft trajectory by yourself and see what happens.

4. Conclusion

In this chapter, we introduced the Kalman filter and extended Kalman filter algorithms. INS/GNSS navigation, target tracking, and terrain-referenced navigation were provided as examples for reader's better understanding of practical usage of the Kalman filters. This chapter will become a prerequisite for other contents in the book for those who do not have a strong background in estimation theory.

Author details

Youngjoo Kim* and Hyochoong Bang
Korea Advanced Institute of Science and Technology, Daejeon, South Korea

*Address all correspondence to: yjkim@ascl.kaist.ac.kr

IntechOpen

© 2018 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

References

[1] Simon D. Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches. Oxford: John Wiley & Sons; 2006

[2] Li XR, Jilkov VP. Survey of maneuvering target tracking. Part I. Dynamic models. IEEE Transactions on Aerospace and Electronic Systems. 2003;**39**(4):1333-1364

[3] Rajamani MR, Rawlings JB. Estimation of the disturbance structure from data using semidefinite programming and optimal weighting. Automatica. 2009;**45**(1):142-148

[4] Matisko P, Havlena V. Noise covariance estimation for Kalman filter tuning using Bayesian approach and Monte Carlo. International Journal of Adaptive Control and Signal Processing. 2013;**27**(11):957-973

[5] Introduction to Kalman Filter and Its Applications. 2018. Available from: https://uk.mathworks.com/matlabcentral/fileexchange/68262-introduction-to-kalman-filter-and-its-applications

[6] Kim Y, Jung W, Bang H. Real-time path planning to dispatch a mobile sensor into an operational area. Information Fusion. 2019;**45**:27-37

[7] Kim Y, Bang H. Vision-based navigation for unmanned aircraft using ground feature points and terrain elevation data. Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering. 2018;**232**(7):1334-1346

[8] Vadlamani AK, de Haag MU. Dual airborne laser scanners aided inertial for improved autonomous navigation. IEEE Transactions on Aerospace and Electronic Systems. 2009;**45**(4):1483-1498

[9] Kim Y, Park J, Bang H. Terrain referenced navigation using an interferometric radar altimeter, NAVIGATION. Journal of the Institute of Navigation. 2018;**65**(2):157-167

[10] Rogers RM. Applied mathematics in integrated navigation systems. American Institute of Aeronautics and Astronautics. 2007

[11] Interpolation. Available from: https://en.wikipedia.org/w/index.php?title=Interpolation&oldid=765887238

[12] Kim Y, Hong K, Bang H. Utilizing out-of-sequence measurement for ambiguous update in particle filtering. IEEE Transactions on Aerospace and Electronic Systems. 2018;**54**(1):493-501