Kalman Filtering and Its Real-Time Applications

Lim Chot Hun, Ong Lee Yeng, Lim Tien Sze and
Koo Voon Chet

Additional information is available at the end of the chapter

http://dx.doi.org/ 10.5772/62352

Abstract

Kalman filter was pioneered by Rudolf Emil Kalman in 1960, originally designed and developed to solve the navigation problem in Apollo Project. Since then, numerous applications were developed with the implementation of Kalman filter, such as applications in the fields of navigation and computer vision's object tracking. Kalman filter consists of two separate processes, namely the prediction process and the measurement process, which work in a recursive manner. Both processes are modeled by groups of equations in the state space model to achieve optimal estimation outputs. Prior knowledge on the state space model is needed, and it differs between different systems. In this chapter, the authors outlined and explained the fundamental Kalman filtering model in real-time discrete form and devised two real-time applications that implemented Kalman filter. The first application involved using vision camera to perform real-time image processing for vehicle tracking, whereas the second application discussed the real-time Global Positioning System (GPS)-aided Strapdown Inertial Navigation Unit (SINU) system implementation using Kalman filter. Detail descriptions, model derivations, and results are outlined in both applications.

Keywords: Kalman filter, real-time, navigation, vehicle tracking, GPS-aided-INS

1. Introduction

Kalman filter exists for the past 50 years. It was first introduced by Rudolf Emil Kalman in 1960 [1] and was implemented on the Apollo Project in 1961 to solve the space navigation problem [2]. Kalman filter is claimed to be an optimal estimator [1] due to its ability to optimally estimate the system's error covariance and use the prediction in a recursive manner to improve the system measurements from time to time. As such, Kalman filter was implemented as the estimator in various applications, such as in navigations [3, 4], image processing [5, 6], and finance [7].

One of the uniqueness in Kalman filter is that it consists of two distinct processes, namely, the prediction process and the measurement process. Both processes are combined and operated in a recursive manner to achieve optimal Kalman filtering process [8]. Another uniqueness of Kalman filter is the incorporation of prediction errors and measurement errors into the overall Kalman filtering process. It is common that each prediction and measurement process consists of errors in random nature. These errors or "noise" are normally being described using the stochastic process. On the other hand, a real-time application can be defined as an application or program that reacts or responses within a predefined time frame, where such predefined time frame is a quantified time using a physical clock [9]. From a real-time application's point of view, the real world's continuous time is turned into discrete time frame Δ. Different real-time applications have different Δ, which in turn defined the response time of the applications. The real-time application must react within the predefined time frame to provide an up-to-date response. Such real-time constraint forced the application to complete its routine within the time frame, else the output may not be accurately reflecting the current state of input [10].

Note that the realization of Kalman filter, in its recursive nature, can be described as a real-time implementation. In this book chapter, the authors will demonstrate two real-time Kalman filtering examples. The first example demonstrated the real-time Kalman filter implementation on vehicle tracking application using vision camera's image processing. A Kalman filtering model is established to estimate the positions and velocities of the moving vehicles and to provide tracking on the vehicles at a normally visible condition [11]. The second example demonstrated the Kalman filter implementation on the real-time Global Positioning System (GPS)-aided Strapdown Inertial Navigation Unit (SINU) System or GPS-aided INU system for Unmanned Aerial Vehicle (UAV) motion sensing. The results obtained from both experiments will be illustrated and discussed in this book chapter.

The outline of this chapter is as follows. Section 2 illustrates the generalized Kalman filter model from real-time system's point of view. Section 3 outlines the real-time vehicle tracking system using vision camera. The contents include the elaboration of image processing algorithms, illustration of the Kalman filtering model on the tracking system, result in acquisition, and discussions. Section 4 depicts the real-time GPS-aided SINU system for UAV motion sensing using Kalman filter. The contents included the derivation of Kalman filter for the GPS-aided SINU system, the offline and real-time implementation of the Kalman filter on the GPS-aided SINU system, results and discussions, and conclusion. Lastly, Section 5 concludes the chapter.

2. Kalman filter

Kalman filtering is a popular technique used to solve observer problems [12] in control engineering [13]. Numerous derivations of the Kalman filter model can be obtained from various researchers' works [3, 8, 12, 14, 15], where detailed elaborations and explanations of the Kalman filter, which included the derivation of the prerequisites such as the state space model and random variables, are outlined. Hence, in this chapter, the authors derived and explained the discrete real-time Kalman filter model from the implementation point of view to ensure readers can understand the idea of Kalman filter from the real-time implementation angle.

2.1. Discrete Kalman filter model

A typical Kalman filtering process is separated into two distinct processes, namely, the prediction process and the measurement process [14]. In general, the Kalman filter prediction model and the measurement model of a real-time system, expressed in discrete form, are as follows:

$$\boldsymbol{x}_k = \phi \boldsymbol{x}_{k-1} + \boldsymbol{Bu} + \boldsymbol{w}_k \tag{1}$$

$$z_k = \boldsymbol{Hx}_k + v_k \tag{2}$$

where x_k is the predicted output, z_k is the measurement output, $\boldsymbol{\phi}$ denotes the state transition matrix, $\boldsymbol{B}$ is the control input matrix, and u is the optional control input matrix. H is the measurement transformation matrix, whereas w_k and v_k are the process noise matrix and measurement noise matrix, respectively. Both Equations (1) and (2) depict the general expression of the Kalman filtering process [14, 15]. In terms of real-time implementation, however, further elaborations are to be performed on Equations (1) and (2).

2.2. Kalman filter algorithm

The Kalman filtering algorithm starts from the prediction process by estimating the prediction state based on the derived state space equation. The state space equation, or state transition equation, may differ in different systems. From the implementation point of view, the expression of the prediction state, similar to Equation (1), is outlined as follows:

$$\tilde{\boldsymbol{x}}_k^- = \phi \tilde{\boldsymbol{x}}_{k-1} + \boldsymbol{Bu} \tag{3}$$

where $\tilde{\boldsymbol{x}}_k^-$ is defined as the *a priori* state estimated at the discrete instant *k*, and $\tilde{\boldsymbol{x}}_k$ is defined as the *a posteriori* state illustrated at the discrete instant *k* given the measurement z_k. Note that, from Equation (3), the *a priori* state $\tilde{x}_k^-$ can be elaborated as a hypothesized state predicted from the system's state transition equations, whereas the *a posteriori* state $\tilde{x}_k$ can be elaborated as the measured state obtained by the system's observation. By letting x_k be the true value of state measurement, the *a priori* prediction error $\boldsymbol{e}_k^-$ and *a posteriori* estimation error $\boldsymbol{e}_k$ can be expressed as:

$$\boldsymbol{e}_k^- = \boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^- \tag{4}$$

$$\boldsymbol{e}_k = \boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k \tag{5}$$

From Equation (4), the *a priori* prediction error covariance can be expressed as:

$$\boldsymbol{P}_k^- = E\left[\boldsymbol{e}_k^- \boldsymbol{e}_k^{-T}\right] = E\left[\left(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^-\right)\left(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^-\right)^T\right] \tag{6}$$

From Equation (6), substituting x_k. Equation (1) and $\tilde{x}_k^-$ with Equation (3) yielded:

$$\begin{aligned}\boldsymbol{P}_k^- &= E\left[\left[\phi\left(\boldsymbol{x}_{k-1} - \tilde{\boldsymbol{x}}_{k-1}\right) + \boldsymbol{w}_k\right] \cdot \left[\phi\left(\boldsymbol{x}_{k-1} - \tilde{\boldsymbol{x}}_{k-1}\right) + \boldsymbol{w}_k\right]^T\right] \\ &= \phi \cdot E\left[\left(\boldsymbol{x}_{k-1} - \tilde{\boldsymbol{x}}_{k-1}\right) \cdot \left(\boldsymbol{x}_{k-1} - \tilde{\boldsymbol{x}}_{k-1}\right)^T\right] \cdot \phi^T + \phi \cdot E\left[\left(\boldsymbol{x}_{k-1} - \tilde{\boldsymbol{x}}_{k-1}\right) \cdot \boldsymbol{w}_k^T\right] + \\ &\quad E\left[\boldsymbol{w}_k \left(\boldsymbol{x}_{k-1} - \tilde{\boldsymbol{x}}_{k-1}\right)^T\right] \cdot \phi^T + E\left[\boldsymbol{w}_k \boldsymbol{w}_k^T\right]\end{aligned} \tag{7}$$

Because the state estimation error and the process noise error are uncorrelated,

$$E\left[\left(\boldsymbol{x}_{k-1} - \tilde{\boldsymbol{x}}_{k-1}\right) \cdot \boldsymbol{w}_k^T\right] = E\left[\boldsymbol{w}_k \left(\boldsymbol{x}_{k-1} - \tilde{\boldsymbol{x}}_{k-1}\right)^T\right] = 0 \tag{8}$$

Therefore, Equation (7) can be simplified into:

$$\begin{aligned}\boldsymbol{P}_k^- &= \phi \cdot E\left[\left(\boldsymbol{x}_{k-1} - \tilde{\boldsymbol{x}}_{k-1}\right) \cdot \left(\boldsymbol{x}_{k-1} - \tilde{\boldsymbol{x}}_{k-1}\right)^T\right] \cdot \phi^T \\ &\quad + E\left[\boldsymbol{w}_k \boldsymbol{w}_k^T\right] = \phi \cdot \boldsymbol{P}_{k-1} \cdot \phi^T + \mathbf{Q}_k\end{aligned} \tag{9}$$

Equation (9) yielded an important step in the prediction process of the Kalman filtering algorithm in obtaining the *a priori* prediction error covariance using the system's state transition matrix $\boldsymbol{\phi}$, the *a posteriori* measurement error covariance from previous estimation $\boldsymbol{P}_{k-1}$ and the process noise covariance $\mathbf{Q}_k = E\left[w_k w_k^T\right]$. Hence, in summary, Equations (3) and (9) summarized the two most important equations in deriving the prediction process of the Kalman filter algorithm.

The next stage of the Kalman filtering algorithm is the measurement process. Equation (2) depicts the observation equation, or the actual measurement equation, of the system. The measurement output z_k is normally obtained from the system's measurement sensors or devices. From here, it is possible to express the *a posteriori* measurement $\tilde{x}_k$ as follows [16]:

$$\tilde{\boldsymbol{x}}_k = \tilde{\boldsymbol{x}}_k^- + \boldsymbol{K}_k\left(\boldsymbol{z}_k - \boldsymbol{H}\tilde{\boldsymbol{x}}_k^-\right) \tag{10}$$

where $\boldsymbol{K}_k$ is the Kalman gain, and the term $\left(z_k - \boldsymbol{H}\tilde{x}_k^-\right)$ is commonly known as the measurement residual or innovation [14–16]. Substituting Equation (2) into Equation (10) yielded:

$$\tilde{\boldsymbol{x}}_k = \tilde{\boldsymbol{x}}_k^- + \boldsymbol{K}_k\left(\boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{v}_k - \boldsymbol{H}\tilde{\boldsymbol{x}}_k^-\right) = \tilde{\boldsymbol{x}}_k^- + \boldsymbol{K}_k\boldsymbol{H}\left(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^-\right) + \boldsymbol{K}_k\boldsymbol{v}_k \tag{11}$$

Given the *a posteriori* measurement error covariance, with reference to Equation (5):

$$\boldsymbol{P}_k = E\left[\boldsymbol{e}_k\boldsymbol{e}_k^T\right] = E\left[\left(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k\right)\left(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k\right)^T\right] \tag{12}$$

Substituting Equation (11) into Equation (12) yielded:

$$\begin{aligned}
\boldsymbol{P}_k &= E\left[\left[\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^- - \boldsymbol{K}_k\boldsymbol{H}\left(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^-\right) - \boldsymbol{K}_k\boldsymbol{v}_k\right]\cdot\left[\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^- - \boldsymbol{K}_k\boldsymbol{H}\left(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^-\right) - \boldsymbol{K}_k\boldsymbol{v}_k\right]^T\right] \\
&= E\left[\left[\left(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}\right)\left(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^-\right) - \boldsymbol{K}_k\boldsymbol{v}_k\right]\cdot\left[\left(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}\right)\left(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^-\right) - \boldsymbol{K}_k\boldsymbol{v}_k\right]^T\right] \\
&= \left(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}\right)\cdot E\left[\left(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^-\right)\cdot\left(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^-\right)^T\right]\cdot\left(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}\right)^T + \boldsymbol{K}_k\cdot E\left[\boldsymbol{v}_k\cdot\boldsymbol{v}_k^T\right]\cdot\boldsymbol{K}_k^T - \\
&\left(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}\right)\cdot E\left[\left(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^-\right)\cdot\left(\boldsymbol{K}_k\cdot\boldsymbol{v}_k\right)^T\right] - E\left[\left(\boldsymbol{K}_k\cdot\boldsymbol{v}_k\right)\left(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^-\right)^T\right]\cdot\left(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}\right)^T
\end{aligned} \tag{13}$$

Because the state estimation error and measurement noise error are uncorrelated,

$$E\left[\left(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^-\right)\cdot\left(\boldsymbol{K}_k\cdot\boldsymbol{v}_k\right)^T\right] = E\left[\left(\boldsymbol{K}_k\cdot\boldsymbol{v}_k\right)\left(\boldsymbol{x}_k - \tilde{\boldsymbol{x}}_k^-\right)^T\right] = 0 \tag{14}$$

Therefore, Equation (13) can be simplified into:

$$\begin{aligned}
\boldsymbol{P}_k &= (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\cdot E\left[\left(x_k - \tilde{x}_k^-\right)\cdot\left(x_k - \tilde{x}_k^-\right)^T\right]\cdot(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})^T + \boldsymbol{K}_k\cdot E\left[v_k\cdot v_k^T\right]\cdot\boldsymbol{K}_k^T \\
&= (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})\cdot\boldsymbol{P}_k^-\cdot(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H})^T + \boldsymbol{K}_k\cdot\boldsymbol{R}_k\cdot\boldsymbol{K}_k^T
\end{aligned} \tag{15}$$

Equation (15) depicts the error covariance update equation in the measurement process of the Kalman filtering algorithm. From Equation (15), one could obtain the optimal Kalman gain K_k

with minimal mean squared error, where the mean squared error is reflected by the *trace* of P_k [16], in which the *trace* is defined as the sum of the diagonal elements in the matrix. To do so, the error covariance update equation from Equation (15) can be rewritten as:

$$\boldsymbol{P}_k = \boldsymbol{P}_k^- - \boldsymbol{K}_k \cdot \boldsymbol{H} \cdot \boldsymbol{P}_k^- - \boldsymbol{P}_k^- \cdot \boldsymbol{H}^T \cdot \boldsymbol{K}_k^T + \boldsymbol{K}_k \cdot \left(\boldsymbol{H} \cdot \boldsymbol{P}_k^- \cdot \boldsymbol{H}^T + \boldsymbol{R}_k\right) \cdot \boldsymbol{K}_k^T \tag{16}$$

The mean squared error reflected by the *trace* of the error covariance P_k can be expressed as:

$$\begin{aligned}\mathbf{T}[\boldsymbol{P}_k] &= \mathbf{T}[\boldsymbol{P}_k^-] - \mathbf{T}[\boldsymbol{K}_k \cdot \boldsymbol{H} \cdot \boldsymbol{P}_k^-] - \mathbf{T}[\boldsymbol{P}_k^- \cdot \boldsymbol{H}^T \cdot \boldsymbol{K}_k^T] + \mathbf{T}[\boldsymbol{K}_k \cdot (\boldsymbol{H} \cdot \boldsymbol{P}_k^- \cdot \boldsymbol{H}^T + \boldsymbol{R}_k) \cdot \boldsymbol{K}_k^T] \\ &= \mathbf{T}[\boldsymbol{P}_k^-] - 2\mathbf{T}[\boldsymbol{K}_k \cdot \boldsymbol{H} \cdot \boldsymbol{P}_k^-] + \mathbf{T}[\boldsymbol{K}_k \cdot (\boldsymbol{H} \cdot \boldsymbol{P}_k^- \cdot \boldsymbol{H}^T + \boldsymbol{R}_k) \cdot \boldsymbol{K}_k^T]\end{aligned} \tag{17}$$

where $\mathbf{T}[\cdot]$ denote the *trace* of matrix and $\mathbf{T}[\boldsymbol{K}_k \cdot \boldsymbol{H} \cdot \boldsymbol{P}_k^-] = \mathbf{T}[\boldsymbol{P}_k^- \cdot \boldsymbol{H}^T \cdot \boldsymbol{K}_k^T]$, as the diagonals of both matrixes are identical. Performing the first derivative of Equation (17) with respect to Kalman gain K_k yielded:

$$\frac{d\mathbf{T}[\boldsymbol{P}_k]}{d\boldsymbol{K}_k} = -2\left[\boldsymbol{H} \cdot \boldsymbol{P}_k^-\right]^T + 2\boldsymbol{K}_k \cdot \boldsymbol{H} \cdot \boldsymbol{P}_k^- \cdot \boldsymbol{H}^T + 2\boldsymbol{K}_k \cdot \boldsymbol{R}_k = 0 \tag{18}$$

where $\frac{d\mathbf{T}[\mathbf{K}_k \cdot \mathbf{H} \cdot \mathbf{P}_k^-]}{d\mathbf{K}_k} = [\mathbf{H} \cdot \mathbf{P}_k^-]^T$ and $\frac{d\mathbf{T}[K_k \cdot (H \cdot P_k^- \cdot H^T + R_k) \cdot K_k^T]}{dK_k} = 2K_k \cdot (\boldsymbol{H} \cdot \boldsymbol{P}_k^- \cdot \boldsymbol{H}^T + \boldsymbol{R}_k)$. Rearranging Equation (18) yielded the optimal Kalman gain with minimal mean squared error, as follows:

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \cdot \boldsymbol{H}^T \cdot \left(\boldsymbol{H} \cdot \boldsymbol{P}_k^- \cdot \boldsymbol{H}^T + \boldsymbol{R}_k\right)^{-1} \tag{19}$$

Lastly, substituting Equation (19) into Equation (16) yielded:

$$\boldsymbol{P}_k = \boldsymbol{P}_k^- - \boldsymbol{P}_k^- \cdot \boldsymbol{H}^T \cdot \left(\boldsymbol{H} \cdot \boldsymbol{P}_k^- \cdot \boldsymbol{H}^T + \boldsymbol{R}_k\right)^{-1} \cdot \boldsymbol{H} \cdot \boldsymbol{P}_k^- = \left(\boldsymbol{I} - \boldsymbol{K}_k \cdot \boldsymbol{H}\right) \cdot \boldsymbol{P}_k^- \tag{20}$$

where Equation (20) is the simplified version of error covariance update equation expressed in terms of optimal Kalman gain obtained from Equation (19) and the *a priori* prediction error covariance obtained from Equation (9).

In summary, the Kalman filtering algorithm can be summarized and is shown in **Figure 1**. The prediction process, as shown in Figure 1, covers the prediction of *a priori* state and *a priori* error covariance. The measurement process, on the contrary, covers the calculation of optimal Kalman gain, updating the *a posteriori* estimation state and the *a posteriori* error covariance. Both processes run in a recursive manner, forming the well-known Kalman filtering algorithm.

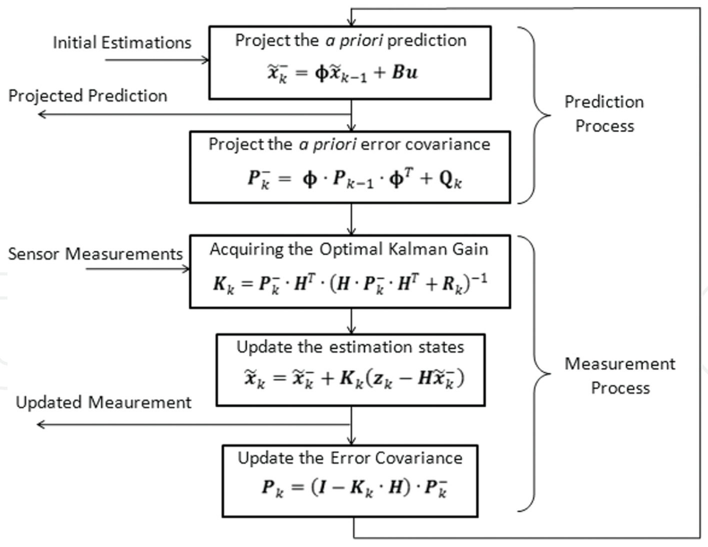
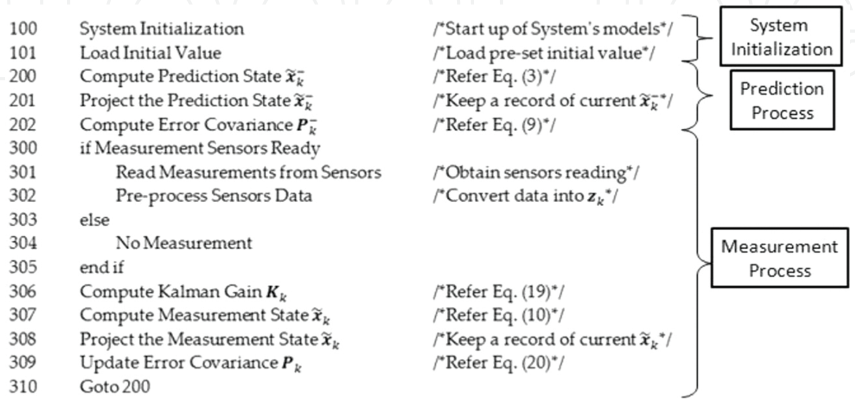

Initial Estimations
Project the *a priori* prediction
$\tilde{x}_k^- = \phi \tilde{x}_{k-1} + Bu$
Projected Prediction
Project the *a priori* error covariance
$P_k^- = \phi \cdot P_{k-1} \cdot \phi^T + Q_k$
Prediction Process
Sensor Measurements
Acquiring the Optimal Kalman Gain
$K_k = P_k^- \cdot H^T \cdot (H \cdot P_k^- \cdot H^T + R_k)^{-1}$
Update the estimation states
$\tilde{x}_k = \tilde{x}_k^- + K_k(z_k - H\tilde{x}_k^-)$
Updated Meaurement
Measurement Process
Update the Error Covariance
$P_k = (I - K_k \cdot H) \cdot P_k^-$

Figure 1. Block diagram of the Kalman filtering algorithm.

2.3. Real-time consideration of Kalman filter

Figure 1 depicts a typical Kalman filtering process algorithm in its recursive form. Notice from the block diagram that the algorithm processed each stage one by one and rewind back to the initial block for the next cycle of processing. From the real-time perspective, there are certain time critical events that need to be handled within a specific time frame. In this subsection, the time critical events are analyzed and discussed as part of the consideration of real-time Kalman filtering algorithm.

The pseudo code of the Kalman filtering algorithm is outlined in **Figure 2**. It is divided into three sections. The first section denotes the system initialization, and it is covered from steps 100 and 101. The second section is the prediction process section, covered from steps 200 to 202. The third and final section is the measurement process section, covered from steps 300 to 310. Note that the second and third sections run recursively.



```
100  System Initialization                       /*Start up of System's models*/
101  Load Initial Value                          /*Load pre-set initial value*/
200  Compute Prediction State $\tilde{x}_k^-$    /*Refer Eq. (3)*/
201  Project the Prediction State $\tilde{x}_k^-$ /*Keep a record of current $\tilde{x}_k^-$*/
202  Compute Error Covariance $P_k^-$            /*Refer Eq. (9)*/
300  if Measurement Sensors Ready
301      Read Measurements from Sensors          /*Obtain sensors reading*/
302      Pre-process Sensors Data                /*Convert data into $z_k$*/
303  else
304      No Measurement
305  end if
306  Compute Kalman Gain $K_k$                   /*Refer Eq. (19)*/
307  Compute Measurement State $\tilde{x}_k$     /*Refer Eq. (10)*/
308  Project the Measurement State $\tilde{x}_k$ /*Keep a record of current $\tilde{x}_k$*/
309  Update Error Covariance $P_k$               /*Refer Eq. (20)*/
310  Goto 200
```

Figure 2. A typical Kalman filtering algorithm process pseudo code.

Figure 3 depicts the timing diagram of the real-time Kalman filtering algorithm based on the pseudo code illustrated in **Figure 2**. The following observations are obtained by examining **Figure 3**:

1. ΔT_{21} can be defined as the time required for Kalman filter prediction process from steps 200 to 202.
2. ΔT_{32} is defined as the time required for the measurement sensor's data preparation from steps 300 to 305. The time consists of reading the measurement data from the sensors and performs the preprocessing on the data as part of the measurement process preparation. Note that ΔT_{32} may be the most time-consuming factor in Kalman filtering process due to the preprocessing step 302. Depending on the application, the preprocessing of measurement data may require a substantial amount of processing power to complete.
3. ΔT_{43} depicts the time required for the measurement data computation process from steps 306 to 310. Three important parameters were computed within this time frame, namely, the optimal Kalman gain, the measurement states, and the error covariance measurements.
4. The total duration for a single iteration is ΔT_{41}, which is equal to ΔT_{21}+ΔT_{32}+ΔT_{43}. Note that ΔT_{41} shall not exceed Δ, where ΔT is defined as the fixed time-step of iteration. In most Kalman filter applications, ΔT is normally adopted as the sampling duration of incoming data. If the processing time for a single iteration exceeded Δ, then the next prediction will not be accurate.

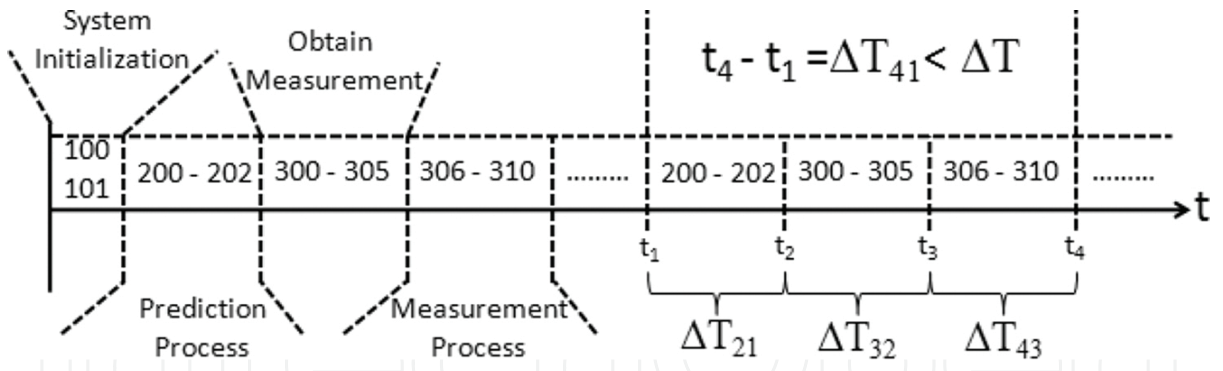


Figure 3. A typical timing diagram of real-time Kalman filtering algorithm process.

3. Vision-based real-time vehicle tracking system

A vision-based real-time vehicle tracking system used vision camera to achieve target tracking [17]. The number of tracked vehicle can be single or multiple. The detection and tracking of vehicles are done through the image processing of consecutive frames of video. Before tracking the vehicles across frames, target detection algorithm such as background subtraction is responsible for isolating the position of the moving vehicles in every frame. The tracking algorithm used the measurements from the detection stage to relate the moving vehicles from frame to frame. However, due to the limitation of performance in the target detection algorithm, it is not reliable to solely depending on the measurements computed from the detection stage. As such, Kalman filtering algorithm can be adopted to compensate the fluctuation and missing measurements whenever the detection stage fails. The missing measurements are predicted based on the center position and velocity of the detected vehicle. An experimental study was conducted on the real-time vehicle tracking at a road junction. The results showed that the Kalman filtering algorithm is capable of tracking the vehicles even with loss measurements appeared on the scene.

3.1. Preprocessing of vision-based vehicle tracking system

Traditionally, the road traffic monitoring is analyzed based on the data collected from the electronic sensor (i.e., loop detector) and manual observation by the human operator. The integration of multiple targets tracking algorithms in the vision-based vehicle tracking system offers an attractive alternative with additional potential to collect a variety of traffic parameters [18]. As illustrated in **Figure 4**, that the vehicle tracking process is the second processing stage in the existing vision-based traffic monitoring system. The performance of this stage greatly depends on the output from the first stage (i.e., the vehicle detection stage). The frames from the video input are recognized as image sequences and fed into the first stage of traffic monitoring system. Moving vehicles are segmented from the stationary background. Because a moving vehicle is formed by a sequence of images from the consecutive frames, the foreground images are matched and combined into its respective tracked objects.

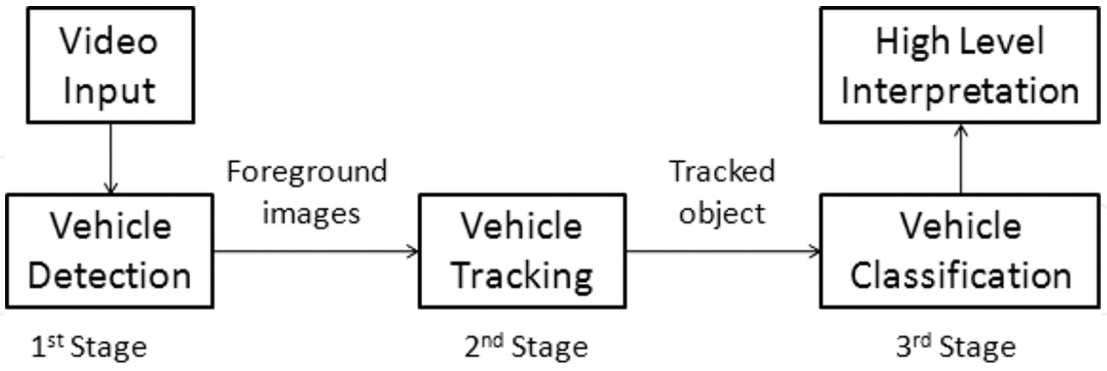


Figure 4. Processing stages of the vehicle monitoring system.

Background subtraction technique is the most widely used image processing algorithm for moving vehicle segmentation [19]. The basic idea of background subtraction algorithm is to subtract (in pixel-wise) all consecutive frames from an occupied background frame. As a result, this algorithm can be easily affected by sudden changes in background and illumination. Since then, numerous researches on updating the background image have been carried out to create a more adaptive background model. However, the contribution effort is still not able to perform vehicle segmentation perfectly, which indirectly affects the performance of vehicle tracking. This is where the Kalman filtering algorithm comes into the picture to improve the tracking performance.

3.2. Kalman filter model for vehicle tracking system

From the vehicle tracking system's point of view, the Kalman filter is to be designed to have the ability to predict the movement of vehicles in the future video frames. The prediction provides a suitable area for searching vehicles in the future frames. Consequently, it shortens the processing time by excluding the foreground images that is not located in the search area [14]. Besides, it also assists the tracking process in the situations where vehicles are temporarily lost due to failed detection.

In the common road traffic flow, vehicle movements can be sufficiently recorded with an optical sensor (i.e., camera) of 25 frames per second. This is because the changes in displacement of moving vehicles in x- and y-positions have been monitored to be small and do not show drastic changes, even at the road junction [20]. Kalman filter can be adopted for predicting the position of the vehicle, particularly during the loss of measurement detections. As discussed previously in Section 2.2, the Kalman filter is a recursive model between the prediction process and the measurement process. The prediction model of the vision-based vehicle tracking system, expressed in real-time discrete form, is outlined as follows:

$$\tilde{\mathbf{x}}_{n,k}^{-} = \left[\tilde{p}_{n,x,k}^{-} \; \tilde{p}_{n,y,k}^{-} \; \tilde{v}_{n,x,k}^{-} \; \tilde{v}_{n,y,k}^{-} \; \tilde{a}_{n,x,k}^{-} \; \tilde{a}_{n,y,k}^{-} \right]^{T} = \phi \cdot \mathbf{x}_{n,k-1} \tag{21}$$

with

$$\phi = \begin{bmatrix} 1 & 0 & \Delta T & 0 & (\Delta T)^2/2 & 0 \\ 0 & 1 & 0 & \Delta T & 0 & (\Delta T)^2/2 \\ 0 & 0 & 1 & 0 & \Delta T & 0 \\ 0 & 0 & 0 & 1 & 0 & \Delta T \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{22}$$

where $[\tilde{p}_{\mathrm{n,x,k}}, \; \tilde{p}_{\mathrm{n,y,k}}]$ denote the predicted vehicle location in pixels, $[\tilde{v}_{n,x,\mathrm{k}}, \tilde{v}_{n,y,\mathrm{k}}]$ denote the predicted velocities of the vehicle in pixels per second, $[\tilde{a}_{n,x,\mathrm{k}}, \tilde{a}_{n,y,\mathrm{k}}]$ denote the predicted vehicle acceleration, and ΔT and n are the sampling instant between image frames and the detected vehicle number, respectively. The predicted error covariance can be calculated as:

$$\boldsymbol{P}_{k}^{-} = \phi \cdot \boldsymbol{P}_{k-1} \cdot \phi^{T} + \mathbf{Q}_{k} = \phi \cdot \boldsymbol{P}_{k-1} \cdot \phi^{T} + \mathbf{Q}_{k} + E\left[\boldsymbol{w}_{k} \boldsymbol{w}_{k}^{T} \right] \tag{23}$$

where $\mathbf{Q}_k$ is assumed to be Gaussian noise of prediction process.

On the contrary, the measurement model of the vision-based vehicle tracking system, expressed in terms of real-time algorithm, is outlined as follows:

$$\mathbf{z}_{n,k} = \mathbf{H} \cdot \mathbf{x}_{n,k} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_{n,x,k} \\ p_{n,y,k} \\ v_{n,x,k} \\ v_{n,y,k} \\ a_{n,x,k} \\ a_{n,y,k} \end{bmatrix} \tag{24}$$

where $[p_{n,x,k},\ p_{n,y,k}]^T$ is the measured vehicle position in pixels, $[v_{n,x,k},\ v_{n,y,k}]^T$ is the measured vehicle velocity in pixels per second, and $[a_{n,x,k},\ a_{n,y,k}]^T$ is the measured vehicle acceleration in pixels per square second. The measured velocity and acceleration can be expressed as:

$$v_{n,x,k} = \left(p_{n,x,k} - p_{n,\mathrm{x},\mathrm{k}-1}\right) / \Delta T \tag{25a}$$

$$v_{n,y,k} = \left(p_{n,y,k} - p_{n,y,\mathrm{k}-1}\right) / \Delta T \tag{25b}$$

$$a_{n,x,k} = \left(p_{n,x,k} - 2p_{n,x,k-1} + p_{n,x,k-2}\right) / \left(\Delta T\right)^2 \tag{26a}$$

$$a_{n,y,k} = \left(p_{n,y,k} - 2p_{n,y,k-1} + p_{n,y,k-2}\right) / \left(\Delta T\right)^2 \tag{26b}$$

With Equations (24) to (26), the optimal Kalman gain can thus be derived using Equation (19) followed by the updates of the estimation state variables $\tilde{\mathbf{x}}_{n,k}$ using Equation (10) and the updates of error covariance P_k using Equation (20). Note that the measurement noise covariance matrix R_k is assumed to be Gaussian noise.

3.3. Experiments and results

An experimental study was carried out using the Kalman filtering model derived in Section 3.2 for real-time vehicle tracking. The experiment used the video stream captured from a static camera installed on a pedestrian bridge above the road, somewhere near the Multimedia University, Melaka, Malaysia. The Kalman filtering model is implemented with C++ programming language. The Open source Computer Vision (OpenCV) library [21] is used for the vehicle detection stage using the background subtraction method based on adaptive Gaussian mixture model in OpenCV. The tracking stage is demonstrated with the Kalman filtering algorithm for associating the foreground images with tracked vehicles from the previous frame.

Figure 5 depicts one of the image frames of the experiment. During the experiment, the tracking of vehicle number 8 (**Figure 5**, left) suffered lost detections due to the imperfection of background subtraction technique. **Figure 6** illustrates the tracking results comparison in terms of x- and y-positions in the image frames for the experiment. Note that there were lost detections in position of vehicle number 8 from frames 190 to 197, as shown in **Figure 6**. Notice that, despite the loss measurements of vehicle number 8, the Kalman filter algorithm can still provide adequate estimations to the vehicle's positions. Note that, although vehicle number 8 was not moving in a straight line, the tracking process was able to update the prediction according to the computed velocity and acceleration from the measurement model. This result shows that the Kalman filtering algorithm assures the continuous tracking of vehicles, although there are several lost measurements during the process.

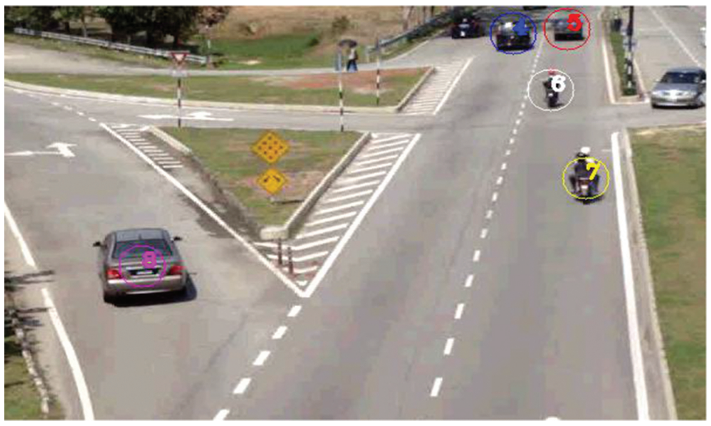


Figure 5. Illustration of one of the image frames of Experiment 2.

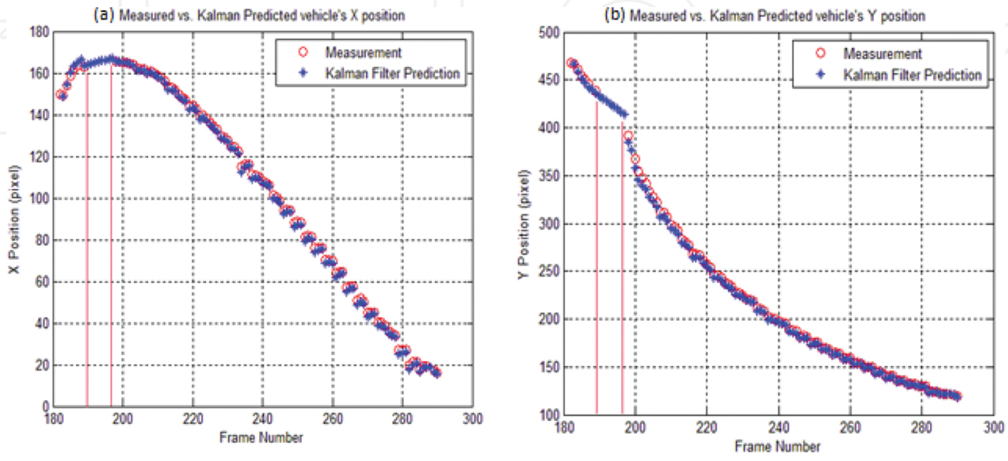


Figure 6. Comparison of vehicle number 8's tracking results for (a) x-position and (b) y-position in image frame.

3.4. Conclusion

This section demonstrated the experimental study of the Kalman filter model for multiple vehicle tracking. The model has incorporated the measurements of center positions of moving vehicles together with the computed velocity and acceleration from the displacement changes in the prediction phase. The tracking results show that the derived Kalman filter model is suitable for tracking multiple vehicles, although measurements are lost in a short period of time.

4. Real-time GPS-aided INU system

Inertial navigation system, which relied on inertial sensors [22] to operate, existed for the past few decades for navigation applications. The SINU is a low-cost inertial sensor developed to substitute the high-cost, high-performance inertial sensors. High-performance inertial sensors are commonly being controlled by government regulations, resulting in unattainable of the sensors in civilian applications. On the contrary, the low-cost, low-performance SINU sensors can be easily acquired, but its measurement data suffered from various errors [23] that jeopardized its accuracy. Due to this issue, the GPS data are adopted as an external reference source to minimize the SINU's errors through the implementation of the Kalman filter.

A typical SINU consists of three orthogonally aligned accelerometers and three orthogonally aligned gyroscopes that provide direct measurement on 3 degrees-of-freedom (DOF) accelerations and 3-DOF angular velocities. Some SINU consists of extra three orthogonally aligned magnetometers for true north measurement. These measurements, as discussed previously, are not accurate. To increase the SINU's accuracy, the GPS's position data obtained from dead reckoning technique is fused with the SINU data through the Kalman filtering algorithm. The system that used such fusion technique is commonly known as the GPS-aided SINU system, in which this fusion is known to retain the advantages of both SINU and GPS while discarding the disadvantages [4].

4.1. Inertial navigation equations

The GPS-aided SINU system is supposed to provide outputs in terms of position, velocity, and orientation. However, the direct outputs provided by the SINU are in terms of accelerations and angular velocities. Hence, the inertial navigation equations, or navigation equations, are formulated to describe the relationship between the GPS-aided SINU system's outputs in terms of the accelerations and angular velocities.

The general form of navigation equations can be derived as follows:

$$\tilde{\mathbf{x}}_k = \begin{bmatrix} \tilde{\mathbf{p}}_k^n \\ \tilde{\mathbf{v}}_k^n \\ \tilde{\mathbf{R}}_{b,k}^n \end{bmatrix} = \begin{bmatrix} \Delta T \cdot \mathbf{v}_{k-1}^n + \mathbf{p}_{k-1}^n \\ \Delta T \cdot \mathbf{R}_{b,k}^n \cdot \mathbf{s}_{k-1}^b + \mathbf{g}^n - 2\mathbf{\Omega}_{ie}^n \cdot \mathbf{v}_{k-1}^n + \mathbf{v}_{k-1}^n \\ \mathbf{R}_{b,k-1}^n \cdot e^{(\mathbf{\Omega}_{ib}^b - \mathbf{\Omega}_{in}^b) \cdot \Delta T} \end{bmatrix} \tag{27}$$

where ΔT represents the sampling time instant of the SINU sensor, $\mathbf{p}_k^n = [p_{x,k}^n \quad p_{y,k}^n \quad p_{z,k}^n]^T$ and $\mathbf{v}_k^n = [v_{x,k}^n \quad v_{y,k}^n \quad v_{z,k}^n]^T$ are the three-dimensional position and velocity in *navigation frame* (or *n-frame*), $\mathbf{R}_{b,k}^n$ represents the direct-cosine-matrix (DCM) that transforms *body frame* (or *b-frame*) to *n-frame*, $\mathbf{s}_k^b = [s_{x,k}^n \quad s_{y,k}^n \quad s_{z,k}^n]^T$ and $\mathbf{g}^n = [0 \quad 0 \quad -9.80665]^T$ represent the three-dimensional acceleration measurement (from the accelerometers) in *b-frame* and the gravitational force in *n-frame*, $\mathbf{\Omega}_{ie}^n$ is the skewed rotation rate matrix in *earth frame* [or *e-frame* with respect to *inertia-frame* (or *i-frame*)] projected to *n-frame*, $\mathbf{\Omega}_{ib}^b$ is the skew matrix of angular velocity measurement (from the gyroscopes) in *b-frame* with respect to *i-frame* projected to *b-frame*, and $\mathbf{\Omega}_{in}^b$ is the skewed transport rate matrix in *n-frame* with respect to *i-frame* projected to *b-frame*. Note that the definition of different *frames* can be found in [4].

4.2. Dynamic error model of inertial navigation equations

The navigation equations outlined in Equation (27) served as the ideal equations to calculate the position, velocity, and orientation based on the data measured from the SINU. However, as discussed earlier, the measurement data from the low-cost SINU contained various dynamic errors that were not reflected in Equation (27). Hence, perturbation process is applied on the navigation equations to acquire the dynamic error equations.

The dynamic error equations, expressed in continuous time, can be elaborated as:

$$\delta\dot{\mathbf{x}} = \begin{bmatrix} \delta\dot{\mathbf{p}}^n \\ \delta\dot{\mathbf{v}}^n \\ \dot{\boldsymbol{\varepsilon}}^n \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{pp} \cdot \delta\mathbf{r}^n + \mathbf{A}_{pv} \cdot \delta\mathbf{v}^n \\ \mathbf{A}_{vp} \cdot \delta\mathbf{r}^n + \mathbf{A}_{vv} \cdot \delta\mathbf{v}^n + (\mathbf{s}^n \times)\boldsymbol{\varepsilon}^n + \mathbf{R}_b^n \cdot \delta\mathbf{s}^b \\ \mathbf{A}_{ep} \cdot \delta\mathbf{r}^n + \mathbf{A}_{ev} \cdot \delta\mathbf{v}^n - (\boldsymbol{\omega}_{in}^n \times)\boldsymbol{\varepsilon}^n - \mathbf{R}_b^n \cdot \delta\boldsymbol{\omega}_{ib}^b \end{bmatrix} \tag{28}$$

where $\mathbf{A}_{pp}$, $\mathbf{A}_{pv}$, $\mathbf{A}_{vp}$, $\mathbf{A}_{vv}$, $\mathbf{A}_{ep}$, and $\mathbf{A}_{ev}$ represent the Jacobians of position, velocity, and orientation error equations [24], respectively, with the subscripts p, v, and e representing the position, velocity, and orientation, respectively. A full description and elaboration of the Jacobians can be found in [24]. $\boldsymbol{\varepsilon}^n$ denotes the orientation errors, the $(* \times)$ operator represents the matrix's cross-product, $\mathbf{s}^n$ and $\boldsymbol{\omega}_{in}^n$ denote the three-dimensional acceleration measurement in *n-frame* and three-dimensional angular velocity measurement in *n-frame* with respect to *i-frame* projected in *n-frame*, $\delta\mathbf{s}^b$ and $\delta\boldsymbol{\omega}_{ib}^b$ denote the three-dimensional acceleration measurement errors in *b-frame* and three-dimensional angular velocity measurement error in *b-frame* with respect to *i-frame* projected in *b-frame*. Note that both $\delta\mathbf{s}^b$ and $\delta\boldsymbol{\omega}_{ib}^b$ are the random errors that reside in the SINU that causes the inaccuracy of the sensor [23].

Equation (28) can be expressed in the state space model as follows:

$$\delta\dot{\mathbf{x}} = \mathbf{A}\cdot\delta\mathbf{x} + \mathbf{B}\cdot\mathbf{u} \tag{29}$$

with

$$\mathbf{A}=\begin{bmatrix}\mathbf{A}_{pp} & \mathbf{A}_{pv} & 0_{3\times3}\\ \mathbf{A}_{vp} & \mathbf{A}_{vv} & (\mathbf{s}^n\times)\\ \mathbf{A}_{ep} & \mathbf{A}_{ev} & -(\boldsymbol{\omega}_{in}^n\times)\end{bmatrix},\ \mathbf{B}=\begin{bmatrix}0_{3\times3} & 0_{3\times3}\\ \mathbf{R}_b^n & 0_{3\times3}\\ 0_{3\times3} & -\mathbf{R}_b^n\end{bmatrix},\ \delta\mathbf{x}=\begin{bmatrix}\delta\mathbf{r}^n\\ \delta\mathbf{v}^n\\ \boldsymbol{\varepsilon}^n\end{bmatrix},\ \mathbf{u}=\begin{bmatrix}\delta\mathbf{s}^b\\ \delta\boldsymbol{\omega}_{ib}^b\end{bmatrix} \tag{30}$$

where $\mathbf{0}_{3\times3}$ is a three-by-three zero matrix. It should be noted that the error matrix u in Equation (30b) can be modeled using the Gauss-Markov model or through the Allan variance analysis [23].

4.3. Kalman filtering model of GPS-aided SINU

The Kalman filter prediction stage of GPS-aided SINU system used the state space model of the dynamic error equations of the SINU to predict the errors. For real-time implementation, the dynamic error equations stated in Equation (29) are to be transformed into its discrete form as follows:

$$\delta\tilde{\mathbf{x}}_k = \Phi\cdot\delta\mathbf{x}_{\mathbf{k-1}} + \mathbf{w}_{k-1} \tag{31}$$

where $\mathbf{\Phi}$ denotes the transition matrix approximated to:

$$\mathbf{\Phi}=\mathbf{I}_{9\times9}+\mathbf{A}\cdot\Delta T=\begin{bmatrix}\mathbf{I}_{3\times3}+\mathbf{A}_{pp}\cdot\Delta T & \mathbf{A}_{pv}\cdot\Delta T & 0_{3\times3}\\ \mathbf{A}_{vp}\cdot\Delta T & \mathbf{I}_{3\times3}+\mathbf{A}_{vv}\cdot\Delta T & (\mathbf{s}^n\times)\cdot\Delta T\\ \mathbf{A}_{ep}\cdot\Delta T & \mathbf{A}_{ev}\cdot\Delta T & \mathbf{I}_{3\times3}-(\boldsymbol{\omega}_{in}^n\times)\cdot\Delta T\end{bmatrix} \tag{32}$$

and $\mathbf{w}_{k-1}$ is the error covariance matrix expressed as:

$$E\left[\mathbf{w}_k\cdot\mathbf{w}_i^T\right]=\begin{cases}\mathbf{Q}_k, & i=k\\ 0, & i\neq k\end{cases} \tag{33}$$

where $\mathbf{Q}_k$ represents the process noise covariance.

The observation equations of the dynamic errors, on the contrary, can be modeled as follows:

$$\mathbf{z_k} = \mathbf{H}_k \cdot \delta\mathbf{x}_k + \mathbf{e}_k \tag{34}$$

with

$$\delta\mathbf{x}_k = \begin{pmatrix} \tilde{\mathbf{p}}_k^n - \mathbf{p}_{GPS,j}^n \\ \tilde{\mathbf{v}}_k^n - \mathbf{v}_{GPS,j}^n \\ \tilde{\vartheta}_k^n - \vartheta_{MEAS,k}^n \end{pmatrix} \tag{35}$$

where $\tilde{\mathbf{p}}_k^n$, $\tilde{\mathbf{v}}_k^n$ and $\tilde{\boldsymbol{\vartheta}}_k^n$ denote the three-dimensional position, velocity, and orientation vectors calculated from the navigation equations, expressed in *n-frame*. On the contrary, $\mathbf{p}_{GPS,j}^n$, $\mathbf{v}_{GPS,j}^n$ and $\boldsymbol{\vartheta}_{MEAS,k}^n$ denote the three-dimensional position, velocity, and orientation vectors obtained from sensors measurement. The variables with subscript GPS indicate the parameters obtained from GPS measurements, whereas the orientation vector $\boldsymbol{\vartheta}\tilde{}_k^n$ can be obtained from the DCM of $\tilde{\mathbf{R}}_{b,k}^n$ [25]. Meanwhile, the orientation measurement vector $\boldsymbol{\vartheta}_{MEAS,j}^n$ is derived as:

$$\vartheta_{MEAS,k}^n = \begin{pmatrix} \phi_{MAG,k}^n \\ \theta_{MAG,k}^n \\ \psi_{MAG,k}^n \end{pmatrix} = \begin{pmatrix} -\text{Atan2}\left(s_{y,k}^n, \sqrt{\left(s_{x,k}^n\right)^2 + \left(s_{z,k}^n\right)^2}\right) \\ -\text{Atan2}\left(s_{x,k}^n, \sqrt{\left(s_{y,k}^n\right)^2 + \left(s_{z,k}^n\right)^2}\right) \\ -\text{Atan2}\left(\psi_1, \psi_2\right) \end{pmatrix} \tag{36}$$

with

$$\begin{aligned} \psi_1 = {} & m_x^n cos\left(\theta_{MAG,k}^n\right) + m_y^n sin\left(\phi_{MAG,k}^n\right) sin\left(\theta_{MAG,k}^n\right) \\ & - m_z^n cos\left(\phi_{MAG,k}^n\right) cos\left(\theta_{MAG,k}^n\right) \end{aligned} \tag{37a}$$

$$\psi_1 = m_y^n cos\left(\phi_{MAG,k}^n\right) + m_z^n sin\left(\phi_{MAG,k}^n\right) \tag{37b}$$

where Equations (36) and (37) represent the orientation measurement vectors. Note that the vector $\left[m_x^n \quad m_y^n \quad m_z^n\right]^T$ refers to the three-dimensional magnetic field strength in *n-frame* obtained from the magnetometers, and the operator atan2 represents the four-quadrant inverse tangent function.

Notice from Equation (35) that there are two different discrete instants described by subscripts *j* and *k*. These two different subscripts described two different discrete instants, with subscript *k* depicts the SINU's discrete instant, whereas subscript *j* depicts the GPS's discrete instant. In most cases, the SINU's sampling rate is much faster than the GPS's sampling rate. This creates phenomena in GPS-aided SINU system's Kalman filtering process that the filter will need to predict a multiple number of predictions before obtaining a measurement from GPS to correct the errors. **Figure 7** illustrates the time operation diagram of the matching of 5 Hz GPS data with the 40 Hz SINU data.

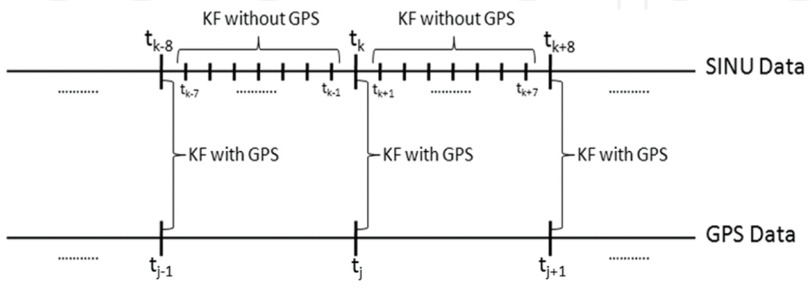


Figure 7. Real-time operational diagram of the GPS-aided SINU system.

4.4. GPS-aided SINU system design and its real-time implementation

Figure 8 delineates the operational block diagram of the design of GPS-aided SINU system. As shown in **Figure 8**, that the SINU consists of three-dimensional accelerometers, gyroscopes, and magnetometers that output three-dimensional accelerations, angular velocities, and magnetic field strengths, respectively. The sampling rate of the SINU is 40 Hz. By combining these data with the GPS data (5 Hz) through the Kalman filtering model, the system is able to compute the estimated position, velocity, and orientation errors, which could be used to improve the overall estimations.

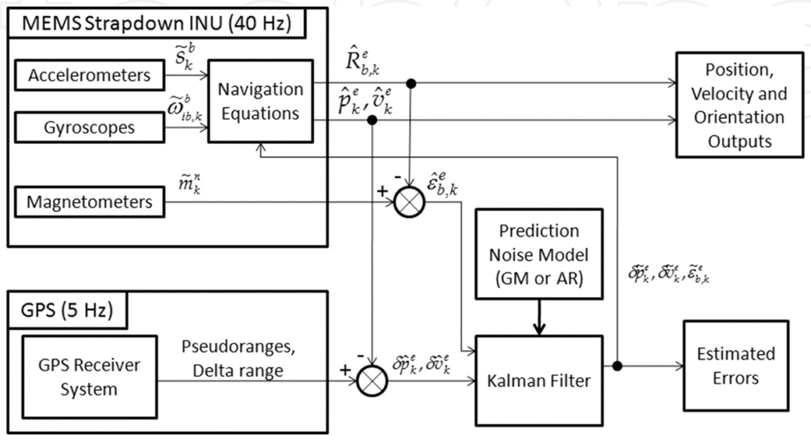
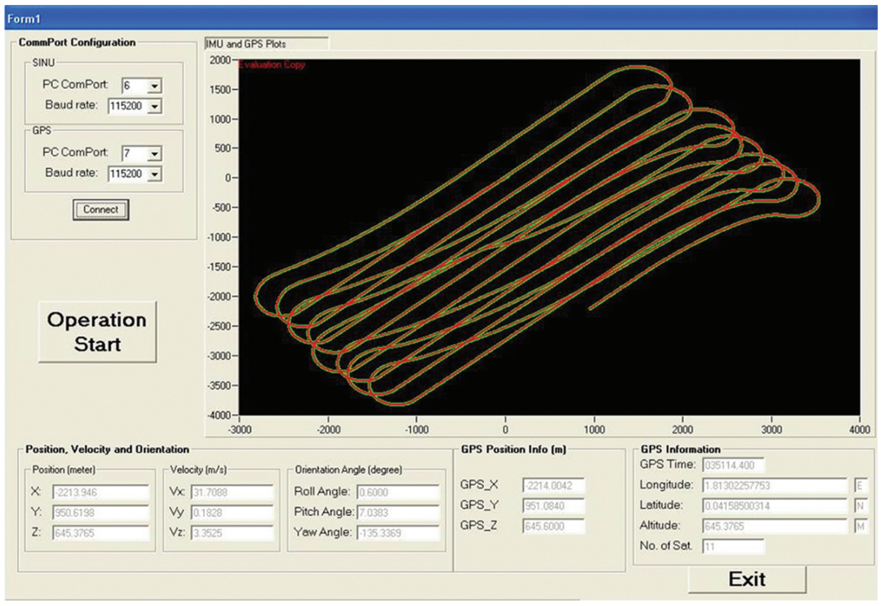

MEMS Strapdown INU (40 Hz)
Accelerometers
Gyroscopes
Magnetometers
$\tilde{s}_k^b$
$\tilde{\omega}_{ib,k}^b$
$\tilde{m}_k^n$
Navigation Equations
$\hat{R}_{b,k}^e$
$\hat{p}_k^e, \hat{v}_k^e$
Position, Velocity and Orientation Outputs
$\hat{\varepsilon}_{b,k}^e$
Prediction Noise Model (GM or AR)
$\delta\tilde{p}_k^e, \delta\tilde{v}_k^e, \tilde{\varepsilon}_{b,k}^e$
GPS (5 Hz)
GPS Receiver System
Pseudoranges, Delta range
$\delta\hat{p}_k^e, \delta\hat{v}_k^e$
Kalman Filter
Estimated Errors

Figure 8. GPS-aided SINU system operational block diagram.

Offline field experiment using a moving car was carried out to verify the performance of the GPS-aided SINU system before real-time implementation. The Kalman filtering process is carried out in offline mode to verify the performance of the developed system. Note that, in the experiment, the SINU and GPS data rate are 40 and 5 Hz, respectively. The data obtained from the offline experiment was fed into the Kalman filtering model to obtain the offline measurements. To test the performance of the proposed GPS-aided SINU system, the same set of offline data was also fed into a conventional GPS-aided SINU system with no magnetometers. The result obtained from the conventional GPS-aided SINU system without magnetometers is compared to the result obtained from the proposed GPS-aided SINU system with magnetometers to verify the performance of the proposed system. From the results of the offline experiment, there is an average difference computed to be 2.84 m between the navigation paths of GPS-aided SINU system with and without magnetometers. The mean difference between the navigation paths of GPS measurements and the GPS-aided SINU system with magnetometers is computed to be ′0.173 m. On the contrary, the mean difference between the navigation paths of GPS measurements and the GPS-aided SINU system without magnetometers is calculated to be ′2.67 m, much higher than the mean difference from the previous calculation. Such results indicate that the proposed system work well in offline mode.

With the success offline implementation on the moving car, the system is now ready for real-time implementation. Both the GPS module and the SINU are connected to an embedded high-performance computer (HPC) through RS-232 for data acquisition and real-time processing. The embedded HPC used in the system come with an Intel Core™ 2 Duo Processor E7500, 4 GB DDR2 RAM, 40 GB hard drive integrated into Zotac Nforce 9300-ITX motherboard. Similar to the previous setting, the SINU's data rate is 40 Hz, which is relatively faster than the GPS data rate of 5 Hz. During the data acquisition stage, the embedded HPC acquired one set of SINU data every 25 ms (equivalent to 40 Hz). On the contrary, the embedded computer acquired one full GPS data every 0.2 s (equivalent to 5 Hz). Hence, it is obvious that there is a mismatch of GPS data to SINU data. As shown in **Figure 7**, when both GPS and SINU data are updated with the newest measurements, the real-time processing system will proceed with the Kalman filtering process to provide a new update on the error prediction. At the instances where newest GPS data were not available, the real-time system will compute the error prediction solely depending on the previous error prediction obtained from the Kalman filtering process and the newest SINU measurements.

A graphical user interface (GUI) is developed using Visual Basic software (from Microsoft Corporation) for the real-time implementation. A total of 31 variables will be saved into the solid-state hard disk continuously in binary file format. The first nine variables represent the raw data from the SINU, which serves the inputs of the GPS-aided SINU system. The subsequent 15 variables represent the computed three-dimensional position, velocity, orientation, acceleration errors, and orientation errors, which serve as the outputs of the GPS-aided SINU system. The last seven variables are the GPS data. **Figure 9** shows the GUI layout and the real-time experimental results of the developed real-time GPS SINU system. **Figure 9** (top left) indicates the serial ports setting for the SINU and the GPS. An"Operation Start"button is located beneath the serial ports frame. An *X*-*Y* graph is used to display both the real-time GPS path in green color line and the real-time SINU's navigation path in red color line. A group of real-time parameters could be found below the *X*-*Y* graph. These parameters included the real-time updated position, velocity, orientation, and GPS information. The incoming raw SINU data are displayed at the bottom of the GUI. Note that the position plots shown in **Figure 9** *X*-*Y* graph are the results from one of the field experiments conducted in Kampung Seri Pantai, Mersing, Malaysia. In this experiment, the GPS-aided SINU system is installed inside an UAV for motion sensing. The navigation path of the UAV, in GPS data, is shown using the Google Earth in Figure 10. The duration of the experiments was approximately 50 min. The recorded average flight speed was 145 km/h.



CommPort Configuration

SINU

PC ComPort: 6

Baud rate: 115200

GPS

PC ComPort: 7

Baud rate: 115200

Position, Velocity and Orientation

Position (meter)		Velocity (m/s)		Orientation Angle (degree)	
X:	-2213.946	Vx:	31.7088	Roll Angle:	0.6000
Y:	950.6198	Vy:	0.1828	Pitch Angle:	7.0383
Z:	645.3765	Vz:	3.3525	Yaw Angle:	-135.3369

GPS Position Info (m)

GPS_X	-2214.0042
GPS_Y	951.0840
GPS_Z	645.6000

GPS Information

GPS Time:	035114.400	
Longitude:	1.81302257753	E
Latitude:	0.04158500314	N
Altitude:	645.3765	M
No. of Sat.	11	

Figure 9. GUI of the real-time GPS-aided SINU system and its field experiment result.

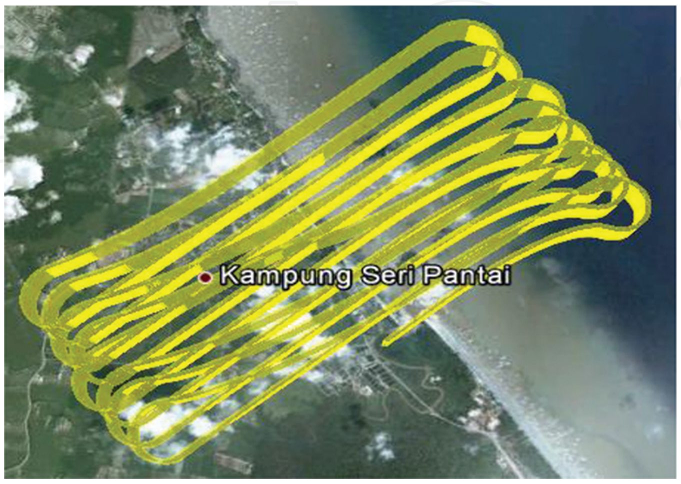


Figure 10. Navigation path of the real-time GPS-aided SINU system experiment on a UAV in Google Earth.

Figures 11 and **12** illustrate the results obtained from the real-time experiment, with **Figure 11** depicting the real-time velocity plot and **Figure 12** depicting the real-time orientation plot. The motion sensing results are compared to the UAV's onboard Piccolo II Autopilot Navigation System [26] outputs. Note that the Piccolo II Autopilot Navigation System is a high-performance, commercial-grade navigation system for UAV autopilot. The mean square differences of position, velocity, and orientation were computed between the developed system and the Piccolo II system and the comparison results are outlined in **Table 1**. Such results indicate that the low-cost GPS-aided SINU system achieved a comparable, adequate performance when compared to a high-performance, high-cost system.

	Mean square difference
Position (m)	0.3081
Velocity (m/s)	0.0077
Orientation (°)	2.5930

Table 1. Mean square difference of position, velocity, and orientation estimation between the proposed GPS-aided SINU system's output with Piccolo II's output.

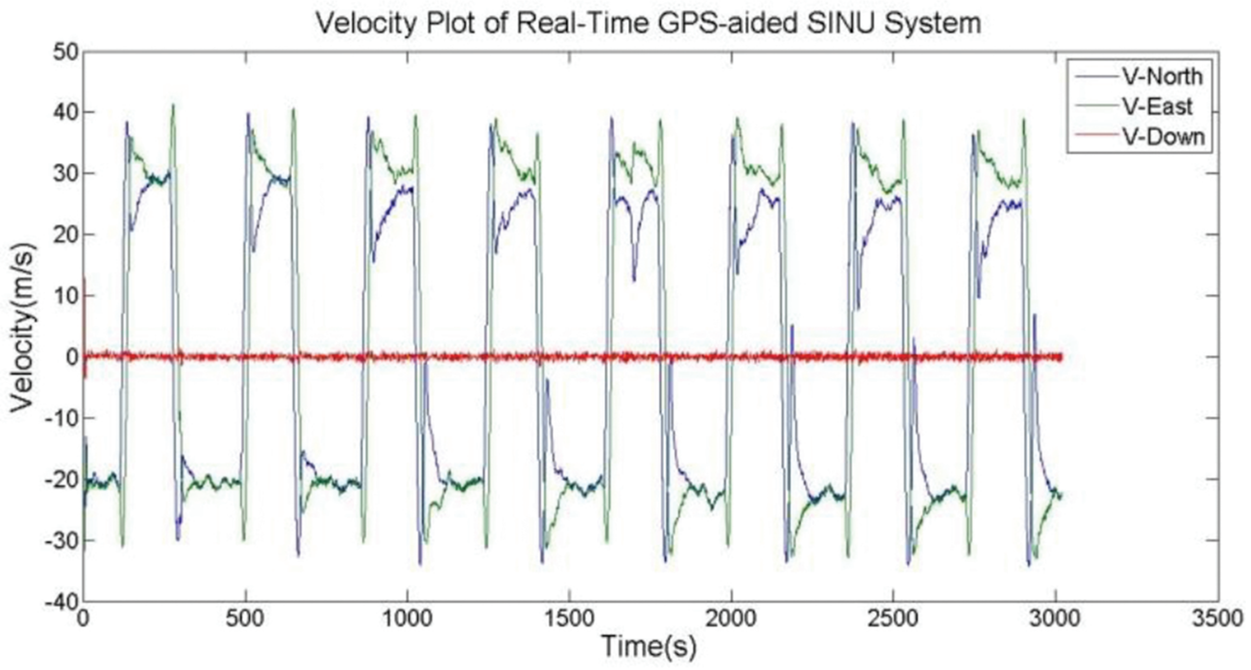


Figure 11. Real-time GPS-aided SINU system velocity plot.

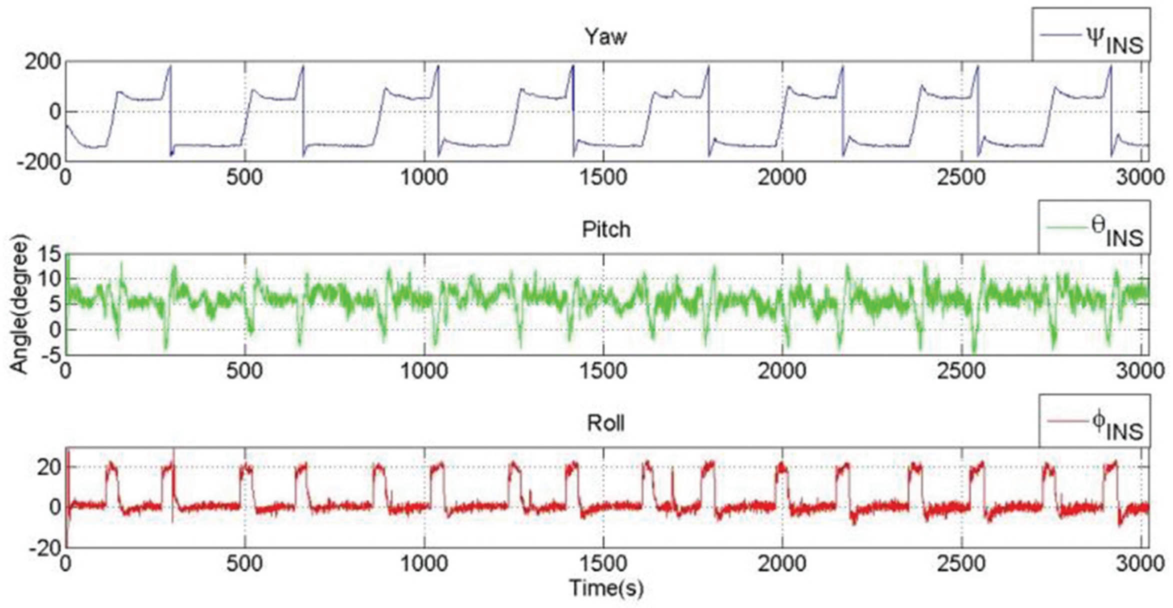


Figure 12. Real-time GPS-aided SINU system orientation plot.

5. Conclusion

This chapter illustrated the real-time implementation of Kalman filter in two applications, namely, the vision-based vehicle tracking system and the GPS-aided SINU system. The Kalman filtering algorithm was derived with the consideration of real-time element. Detail illustrations on deriving the Kalman filtering models for the vision-based vehicle tracking system and the GPS-aided SINU system were outlined and discussed. Both implementations were put on real-time experiments, and the results from both implementations were recorded and analyzed. The results show that the real-time Kalman filtering algorithms work well in the applications.

Author details

Lim Chot Hun[1*], Ong Lee Yeng[2], Lim Tien Sze[1] and Koo Voon Chet[1]

*Address all correspondence to: chlim@mmu.edu.my

1 Faculty of Engineering & Technology, Multimedia University, Melaka, Malaysia

2 Faculty of Information Science & Technology, Multimedia University, Melaka, Malaysia

References

[1] Kalman, R. E. A new approach to linear filtering and prediction problems. Trans ASME J Basic Eng. 1960;82(Series D):35–45.

[2] Grewal, M. S., Andrews, A. P. Applications of Kalman filtering in aerospace 1960 to the present [historical perspectives]. IEEE Control Syst Mag. 2010;30(3):69–78. DOI: 10.1109/MCS.2010.936465

[3] Barczyk, M., Lynch, A. F. Invariant observer design for a helicopter UAV aided inertial navigation system. IEEE Trans Control Syst Technol. 2013;21(2):791–806. DOI: 10.1109/TCST.2012.2195495

[4] Chot, H. L., Tien, S. L., Voon, C. K. Design and development of a real-time GPS-aided SINU system. Int J Adv Robot Syst. 2012;9:1–9. DOI: 10.5772/52681

[5] Salti, S., Lanza, A., Di Stefano, L. Synergistic change detection and tracking. IEEE Trans Circuits Syst Video Technol. 2015;25(4):609–622. DOI: 10.1109/TCSVT.2014.2355695

[6] Bresson, G., Feraud, T., Aufrere, R., Checchin, P., Chapuis, R. Real-time monocular SLAM with low memory requirements. IEEE Trans Intell Transport Syst. 2015;16(4): 1827–1839. DOI: 10.1109/TITS.2014.2376780

[7] Racicot, F.-E., Theoret, R. Forecasting stochastic volatility using the Kalman filter: an application to Canadian interest rates and price-earnings ratio. IEB Int J Finance. 2010;1:28–47.

[8] Faragher, R. Understanding the basis of the Kalman filter via a simple and intuitive derivation. IEEE Signal Process Mag. 2012;29(5):128–132. DOI: 10.1109/MSP.2012.2203621

[9] Mall, R. Real-Time Systems: Theory and Practice. 1st ed. Prentice-Hall; 2009. 242 p. ISBN: 81-317-0069-0.

[10] Kopetz, H. Real-Time Systems: Design Principles for Distributed Embedded Applications. 2nd ed. Springer; 2011. 376 p. DOI: 10.1007/978-1-4419-8237-7.

[11] Ong L. Y., Lau, S. H., Koo, V. C., Lim, C. H. An experimental study on vision-based multiple target tracking. Int J Microwave Opt Technol. 2014;9(1):134–138.

[12] Janiszewski, D. Extended Kalman Filter Based Speed Sensorless PMSM Control with Load Reconstruction, Kalman Filter, Vedran Kordic (Ed.), ISBN: 978-953-307-094-0, InTech.

[13] Ogata, K. Modern Control Engineering. 5th ed. Prentice-Hall; 2010.

[14] Welch, G., Bishop, G. An Introduction to the Kalman filter. Department of Computer Science, University of North Carolina, Chapel Hill, Tech. Rep. TR95041; 2000.

[15] Galleani, L., Tavella, P. Time and the Kalman Filter. IEEE Control Syst. 2010;30(2):44–65. DOI: 10.1109/MCS.2009.935568.

[16] Lacey, T. Tutorial: The Kalman filter. Computer vision. Available at: http://web.mit.edu/kirtley/kirtley/binlustuff/literature/control/Kalman%20filter.pdf.

[17] Goo, J., Aggarwal, J. K., Gokmen, M. Tracking and segmentation of highway vehicles in cluttered and crowded scenes. IEEE Workshop on Applications of Computer Vision (IEEE WACV); 2008.

[18] Huang, C. L., Liao, W. C. A vision-based vehicle identification system. Proceedings of the 17th International Conference on Pattern Recognition (ICPR 2004), 2004; 4: 364–367.

[19] Jie, Y., Wang, J. Q., Lu, H. Q. A hierarchical approach for background modeling and moving objects detection. Int J Control Automation Control. 2010;8:940–947.

[20] Aydos, C., Bernhard, H., William, U. Kalman filter process models for urban vehicle tracking. 12th International IEEE Conference on Intelligent Transportation Systems; 2009; 1–8.

[21] Open Source Computer Vision. Available at: http://opencv.org. Accessed 12 April 2013.

[22] Barbour, N., Schmidt, G. Inertial sensor technology trends. IEEE Sensors J. 2001;1(4):332–339.

[23] Lim, C. H., Tan, W. Q., Lim, T. S., Koo, V. C. Practical approach in estimating inertial navigation unit's errors. IEICE Electron Express. 9(8):772–778.

[24] Kong, X. Y. Inertial navigation system algorithms for low cost IMU. Ph.D. thesis, The University of Sydney, August 27, 2000.

[25] David, H. T., John, L. W. Strapdown Inertial Navigation Technology. 2nd ed. The Institution of Electrical Engineering and Technology, United Kingdom; 2004.

[26] Cloud Cap Technology. Piccolo Autopilots.Available at: http://www.cloudcap-tech.com/piccolo_system.shtm. Accessed 15 December 2015.