Compositional Kalman filters for navigational data streams in IoT systems

by

Yuri Boiko

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfilment of the requirements
For the M.Sc. degree in
System Sciences

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

The Internet of Things (IoT) technology is undergoing expansion into different aspects of our life, changing the way businesses operate and bringing in efficiency and reliability of digital controls on various levels. Processing large amount of data from connected sensor networks becomes a challenging task. Specific part of it related to fleet management requires processing of the data on boards of vehicles equipped with multiple electronic devices and sensors for maintenance and operation of such vehicles. Herewith the efficiency of various configurations of employing Kalman filter algorithm for on-the-fly pre-processing of the sensory network originated data streams in IoT systems is investigated. Contextual grouping of the data streams for pre-processing by specialized Kalman filter units is found to be able to satisfy the logistics of IoT system operations. It is demonstrated that interconnection of the elementary Kalman filters into an organized network, the compositional Kalman filter, allows to take advantage of the redundancy of data streams to accomplish IoT pre-processing of the raw data. This includes intermittent data imputation, missing data replacement, lost data recovery, as well as error events detection and correction. Architectures are proposed and tested for the interaction of elementary Kalman filters in detection of GPS outage events and their compensation via data replacement procedure, as well as GPS offset occurrence detection and its compensation via data correction routine. Demonstrated is the efficiency of the suggested compositional designs of elementary Kalman filter networks for the purpose of data pre-processing in IoT systems.

Acknowledgements

I would like to thank all the people who made this possible, particularly my supervisors, Professor Tet H. Yeap and Professor Iluju Kiringa, for their consistent efforts in guiding my research work.

Meaningful interactions and discussions with Professor Riadh W.Y. Habash are gratefully acknowledged, particularly within System Science program's Control System course SYS-5100, in which essential theoretical and positive practical experiences of operating with Kalman filter have been acquired and enabled further progress leading to the research work of this thesis.

Dedication

This is dedicated to the one I love.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Data flow in the fleet management architecture of IoT is determined by the operational conditions of the relevant sensors, as well as by the external network environment. There is typically a significant need for data imputation in IoT scenarios. Among these scenarios, the most significant is due to the need to equalize the measurement data frequency within different groups of sensors, when simultaneous processing of the relevant data is required by the algorithms involved (such as data fusion, clustering, etc.). In this scenario, data imputation consists of supplementing actual measurements for some sensors with intermediate data points consistent with the available data which, although not actual measurements, reflect with maximized likelihood the data values which should have occurred at the specified time points. An alternative scenario requiring data imputation is when measured points are lost in the transfer network. Here, lost data are substituted with artificially generated data points, which are made statistically consistent with the rest of the data. System outages are another example requiring data imputation. This includes GPS signal outages, which interrupt the data flow for a period of time. In this case data imputation would serve as temporary replacement of the data until system recovery, which would restore the data flow operation. Additionally, data errors such as erroneous measurements, and anomalous data such as outliers, also require detection and correction, of which imputation can be the preferred method of choice to ensure continuous data flow in the system. Therefore, the motivation of the current research effort is in identifying the most efficient ways of data imputation when dealing with scenarios of fleet management in the IoT environment. In the next section a more detailed outline of the motivation is presented.

1.2 State of the Art

The data flow in the IoT fleet management architecture is generated by the sensor networks installed on the vehicles. It serves as a ground for various undertakings, such as
(1) decision making on-the-fly, when critical events occur in real-time mode;
(2) data cleaning in the database to ensure consistency of the saved data for future processing via various types of statistical analysis. An obvious example of type (1) is outage of the GPS signal during the operation of systems, such as cars and other vehicles, equipped with sensor networks and connected to the IoT infrastructure. Type (2) scenarios may occur in off-line operations, such as data transfers, database cleaning, statistical analysis, and, in particular, queries based on selected sets of the parameters with expectedly related dependencies. Based on the last scenario, it would be essential to equalize the rates of the data points flow for algorithmic search of hidden correlations. In this case, data imputation becomes a tool for complementing intermediate points within the data flow of the lower data rate to equalize it with another one of the higher data rate.

Offline data imputation procedures are well developed and accumulate significant prior history of statistical algorithms and tools. Rubin (1987) [30] had introduced the procedure of multiple imputations to address the problem of missing values. In essence, his approach delivers a set of plausible values for imputation of missing values via consideration of statistical uncertainty of imputed missing values. This allows compensating for missing data in surveys. Extending multiple imputation procedures to combinations of surveys in multivariate cases was developed by Raghunathan et al (2001) [27] via the employment of a series of regression models and proved to be useful for multiple imputations of cross-sectional data (Raghunathan (2006)) [26]. The effectiveness of the simultaneous use of multiple imputation procedures for multivariate cases had been demonstrated by Reiter (2004) [28], Reiter (2007) [29].

Missing data in the time series had been addressed by Hopke et al (2001) [16] for the purpose of multiple imputations of data points. For this, the authors used the statistical model of integrated moving average, modified to reflect seasonal variations and thus introducing structural features in the time series under processing. Generally speaking, structural components in the definition of the time series aim to reflect the underlying nature of the time series under investigation, which in turn can most comprehensively be described in the state-space model of the time series. While state-space is hidden from observation, the structural components are openly observable. However, they are directly connected to the state-space variables thus revealing part of the hidden underlying nature of the time series to the outside observer. While the state-space model of the time series is not reachable for manipulation as being hidden, the structural model of the time series may be freely changed to better reflect the observed behaviour of the time series. This connection between the structural components of the time series and the state-space model makes it attractive to present the time series in the structural form, as the state space may contain similar structural components, such as trends, seasonality, and autoregressive features (Sohae Oh (2015)) [23].

Research efforts in analyzing time series with state-space model (Durbin and Koopman (2002) [8], West and Harrison (1997) [32], Kunsch (2001) [17], Migon et al (2005) [21]) lead to the multiple imputation models for a time-series cross-section data by Honaker et al (2010) [15]. In that case, the state-space model for time-series was used to reflect the properties of the time-series. The multiple imputation models were constructed there to incorporate unique features of time-series via state-space. As pointed out by West and Harrison (1997) [32], the important advantage of the state-space modelling is its capacity to accommodate multivariate time series, even in non-stationary case (such as random walk and others). The issue of the reliability of the conclusions drawn from non-stationary data is valid and does require special consideration on a case-to-case basis. West and Harrison (1997) [32], considered a multivariate dynamic linear model (DLM) as a linear and Gaussian case of the state-space model. There the state-space variables satisfy linear equations and joint distribution of the state-space variables and observation variables are multivariate Gaussian (i.e. normal).

Prado and West (2010) [25] introduced the exchangeable time series, which represent sets of highly correlated multiple chains of time series, for which univariate dynamic linear models are extendible into multivariate dynamic linear ones due to presumably same or similar state evolution over time.

To accommodate non-Gaussian, non-linear class of state-space models, Carlin et al (1992) [5] proposed engaging sequential state simulations state-by-state. Procedurally, this meant drawing state variables one at a time on the time interval [1, T], i.e. when time t starts from 1 and ends at T, and thus obtaining a latent state vector and covariance matrix for the time t=T. The low convergence to the posterior distribution in Carlins approach prompted Carter and Kohn (1994) [6] and Fruhwirth-Schnatter (1994) [9] to develop a technique called Forward Filtering Backward Sampling (FFBS), which offers improved convergence to the posterior distribution. The FFBS algorithm essentially contains two sequential stages - the initial one, named forward filtering, is followed by another one, called backward sampling. At the forward filtering stage the algorithm sequentially updates variables of the evolution equations for the time sequence t= 1,2, 3, ,T, for which purpose the Kalman filter is employed. After completing full time series, i.e. reaching time t=T, the backward sampling stage of the technique begins, moving in reverse time order t = T, T-1, , 2, 1, and producing sequence of state vectors with related covariance matrices.

State-space accepts the structural models as well as Auto-Regressive Integrated Moving Average (ARIMA) models (Harvey (1989)) [13], Brockwell and Davis (1991) [3] and Hamilton (1994) [12]. Moreover, quite a few structural models may be expressed via ARIMA formalism. Additionally, one important procedure to estimate dynamic systems represented in state-space is the Kalman filter algorithm (Kalman (1960)) [20]. The Kalman filter provides a solution to the problem of linear systems via recursive procedure in discrete time space.

1.3 Problem Formulation

The main goal of this research effort is implementation of pre-processing procedures of the raw data streams relevant to the IoT systems.

So far, available literature favours the Kalman filter algorithm in the areas of data pre-processing, where the combination of speed and accuracy is essential and relevant to IoT scenarios. However, the existing information is not IoT-specific and therefore is not sufficient in terms of targeting application-oriented development, which is in the focus of current trends of IoT research efforts.

Therefore, the stated goal and the tool selection requires verification of the effectiveness of Kalman filter data imputation for specific structural time series models relevant to sensory data flow in IoT fleet management environment. More specifically, the IoT data imputation scenarios need to be identified via comprehensive testing where the Kalman filter algorithm allows materializing of its natural advantages. This also needs to be supplemented with verification of the scenarios of on-the-fly data pre-processing essential for IoT systems. The above arguments justify the objectives outlined in the next section.

1.4 Thesis Objectives

The objectives of this research effort include implementation of the following pre-processing operations to the raw data streams in the IoT environment: (i) noise reduction;
(ii) intermittent data imputation;
(iii) missing data recovery;
(iv) error detection and correction, including outlier replacement.

The Kalman filter is selected as an implementation tool for achieving these objectives. As in the IoT context — the fleet management is selected within the navigational paradigm of the vehicle travelling via a pre-defined route of the constructed path way. The Kalman filter is to be used to collect the data characterizing the vehicle's navigation with the best possible accuracy and with the desired degree of detailing its features, such as the quantization level of its characteristic variables.

The contribution of this research into the IoT field is summarized in the next section, 1.5.

1.5 Thesis Contributions

The contribution to knowledge of this thesis comprises the application of known Kalman filter techniques in the new area of fleet management in the IoT field. The main concept implemented in this thesis consists of the two major steps. First, the complicated task of pre-processing multiple IoT data streams with the unified Kalman filter is decomposed to a set of simpler tasks, namely, pre-processing small groups of inter-related data streams

composed within specified contexts. This de-compositional step is widely used in the Kalman filter techniques and originality here is in its application within the specific IoT context of a fleet management. Secondly, the obtained elementary Kalman filters are grouped on a contextual basis to serve the task of fleet management events detection. The groups of two Kalman filters interfaced for specific events detection are implemented within this research work to verify its operation in IoT applications. This verification has been confined within known navigational paradigms of Kalman filter employment. However, the originality here is in carrying it within the fleet management context, expandable to cover other functionalities of the vehicles of the fleet, for instance those related to the vehicle health, vehicle-driver interactions, and vehicle-to-vehicle information exchange. To emphasize such potential expandability in application, the specialized compositional Kalman filter is introduced in reference to each mini-group of interfaced Kalman filters. In this way, we address the prospects of creating new compositions to meet other needs of the fleet management. Such needs may include predictive maintenance tasks of fleet's vehicles, when the health of vehicles is constantly monitored in various real-life situations to create the best possible conditions for early detection of potential technical problems, thus enabling prevention in an optimal way.

The following compositional Kalman filter architectures are suggested to achieve objectives of this research work.
- Firstly, the Kalman filter employing GPS, orientation and speedometer sensors is architecturally interfaced with a GPS-free Kalman filter relying on orientation and speedometer only. Here, the navigation is carried out in GPS-free mode, while compensating positional error accumulation via GPS pins.
- Secondly, the above architecture is modified to handle GPS outage events.
- Thirdly, further modification of the initial architecture enables GPS offset detection and correction.

The algorithmic procedures to implement the above architectures are developed and embodied in a specialized software in MATLAB programming language. Simulations of fleet management scenarios are conducted at various noise levels for the sensors involved as well as various operational rates. The numerical data are gathered and verified to confirm applicability of the compositional Kalman filter approach for fleet management tasks, including specialized events detection and data flow corrections, thus achieving the objectives set out herewith.

The verification of the usefulness of the Kalman filter approach in IoT scenarios is pursued through modelling and simulation of the navigational circumstances, in which the above mentioned abilities (i) through (iv) of the Kalman filter become useful. The particulars of pursuing the objectives (i) through (iv) are as follows:

- The objective (i), i.e. noise reduction of IoT data streams, is achieved by Kalman filtering of the raw data streams. It is then verified by comparing the navigational data streams produced by the Kalman filter-based observer with its actual noisy original and extracting the essential difference between the two.

- The objective (ii), i.e. imputation of the intermittent data in IoT, is achieved by engaging the compositional Kalman filter for upgrading the GPS data stream rates. Specifically,

initially low rate GPS data streams are converted to higher rate to match that of the data stream from other sensors, such as speedometer.

- The objective (iii) of IoT's missing data recovery is achieved by setting up the compositional Kalman filter to handle GPS outage events. This is when the filter recovers the data lost due to the outage of GPS sources by processing the data from other sensors and eventually replacing the missing data points with sufficient accuracy.

- The objective (iv) of error detection and correction in IoT data streams is achieved by designing a compositional Kalman filter capable of counteracting the GPS offset occurrence events. The specialized compositional Kalman filter has been demonstrated to possess sensitivity sufficient to detect and flexibility to correct errors caused by the offset events, which distorted data flow in the GPS sensors. The particular case of outlier detection and correction here has been demonstrated as the easier case for very large sporadic offsets.

The above outline of this thesis's objectives and scenarios applied to demonstrate their achievement within specific IoT contexts is summarized in Table 3.3 on page 26.

1.6 Thesis Organization

This thesis is organized into five chapters, of which the original work is presented in the final three chapters. Chapter 1, an introduction, describes the motivation of the undertaken research, rooted in the imperfections of the data streams in IoT systems and the resulting needs of such data pre-processing. It outlines existing methods for data processing in IoT field, proposing the Kalman filter concept as potentially beneficial for processing multiple data streams from noisy sensors. It also draws attention to the insufficiency of existing literature for the straightforward, efficient applicability of the Kalman filter technique for processing multiple noisy data streams without contextual organization of the data and modification of the Kalman filter designs to meet IoT-specific requirements. The problem formulation thus follows, and research objectives are set up. The contributions of this thesis are then summarized.

Chapter 2 gives a general background review of existing literature on data stream processing with emphasis on IoT specific tasks, such as data imputation, error correction, and missing data recovery. The potential roles of existing techniques in IoT applications is also considered. The competitive edge of the Kalman filter approach is identified in the on-the-fly pre-processing domain.

In Chapter 3, the concept of the compositional Kalman filter is introduced, which aims to achieve the declared objectives. Interfacing two elementary Kalman filters as architectural solutions for compositional Kalman filter implementation is detailed. Three major architectures are presented: the general one for GPS pinning, and two event-specific architectures — namely, one for GPS outage handling and another for GPS offset detection and correction. The algorithmic solutions for each architecture implementation is given here and its logical constructs are described.

The results of the simulated implementation of the models for the suggested architectures are gathered in Chapter 4. The experimental data are analysed in view of the

objectives put forth earlier. Conclusive arguments are then presented in support of achieving the objectives.

The main conclusions are gathered in Chapter 5, outlining the achieved objectives. Based on that, prospects for possible future research are outlined on further expansion of the concept of the compositional Kalman filter in the IoT domain.

References for used literature sources are provided at the end.

Chapter 2

Related Work

In this chapter, we outline the current status of available approaches to operate with data stream processing, including those with data imputation capabilities viewed as potentially suitable to IoT field. Specific attention is drawn to the classification of the structural time series as a basis for state representation of the data flow from the sensors monitoring state of the IoT systems. This consideration will be extended to the principles of the employment of the Kalman filter in IoT environments.

2.1 Time Series Data Streams

2.1.1 Univariate Time Series

When the state of a system is observed via set of sensors, the data flow generated by each individual sensor is represented by time series values, for example y_t, which in turn are linked to the state variable values of s_t as shown in Figure 2.1 .

Definition of the model based on univariate time series can be expressed by the following equation (see Jalles (2009) [18]):

$$y_t = \mu_t + \psi_t + \nu_t + \varepsilon_t \tag{2.1}$$

where μ_t is the trend (or level), ψ_t is the cycle, ν_t is the seasonal component and ε_t is the irregular component of the contributing random changes, which represent sensor measurement noise.

One of the simplest example of (2.1) is a time series whose observations move irregularly around its average level μ_0 which itself is constant over time:

$$y_t = \mu_0 + \varepsilon_t \tag{2.2}$$

for all $t = 1, 2, ..., (T-1), T$, where ε_t is a Gaussian white noise process with variance σ^2. This is the case of a time series with a deterministic level of the mean component

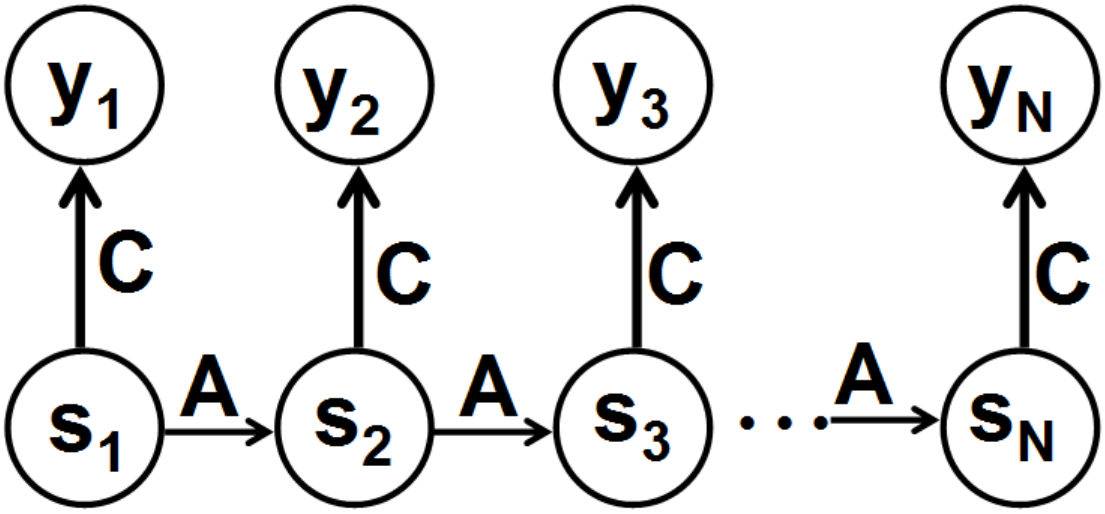


Figure 2.1: Hidden Markov chain representation of structural time series model

μ_0 in the white noise environment. For the fleet management in IoT this may be a GPS-based noisy measurement of the immobile vehicle location. Alternatively, the measured by speedometer value of the vehicle's speed in the cruise control mode on the straight part of the highway (i.e. when motion is uniform and stationary, with no acceleration).

The next level of complexity in the time series given by equation (2.2) would require the replacement of the deterministic portion of the level μ_0 with the time dependent component μ_t. Such a replacement relaxes the deterministic restriction on the level component, thus allowing the level component to change through the time. Structuring of the time series may impose a specific form of time dependence, which converts the time series to the so-called local level model:

$$y_t = \mu_t + \varepsilon_t \tag{2.3}$$

where

$$\mu_t = \mu_{t-1} + \varepsilon_t \tag{2.4}$$

with $\eta_t \sim \text{NID}(0, \sigma_\eta^2)$ functioning as a Gaussian disturbance term, randomly contributing around the underlying level, lending uncertainty to the measurement of the posterior level μ_{t-1} (with no particular bias). Equation (2.4) describes noisy random walk model . The robotic vehicle scanning the area in a random search with no specific target in view, such as in surveillance applications may be a specific practical IoT example of it.

Practical examples of univariate structural time series were reviewed by Moritz et al (2015) [22] and are reproduced herewith in Figure 2.2. Three major components of interest are seen there: (i) trend, (ii) seasonal and (iii) irregular components.

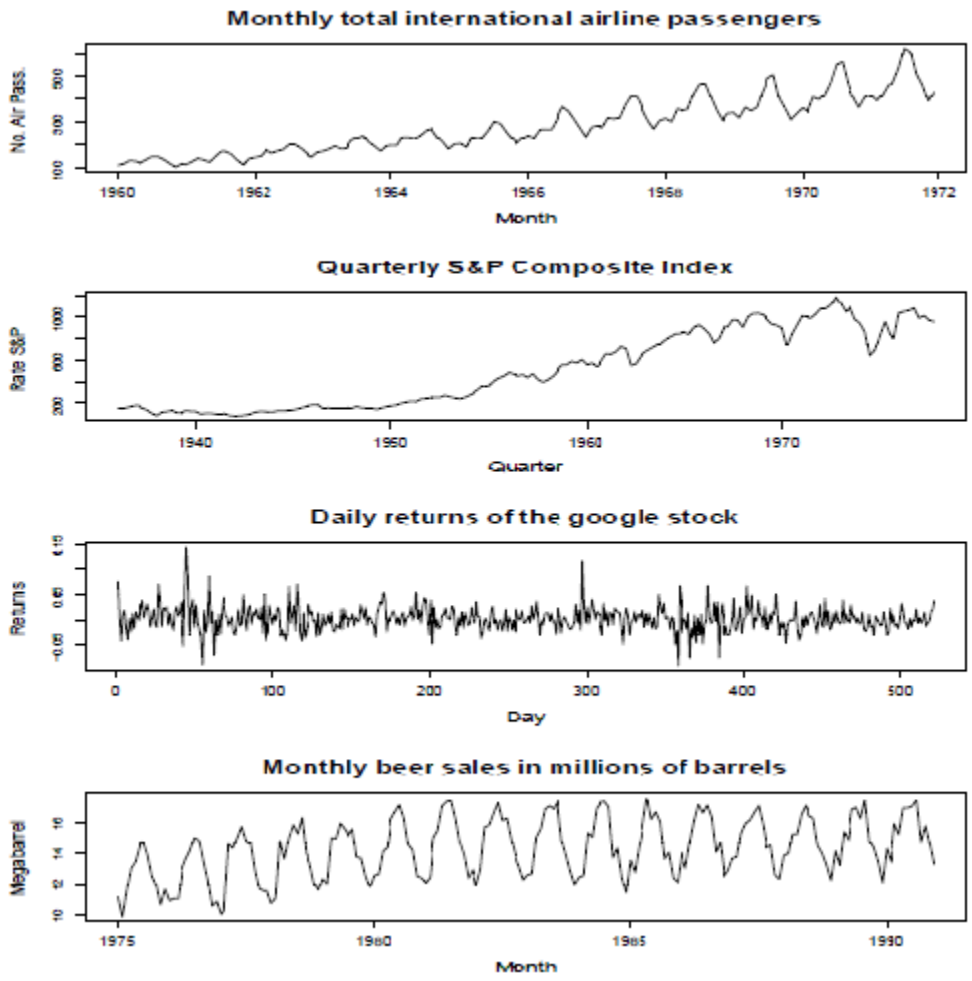


Figure 2.2: Examples of various time series datasets (from Moritz et al (2015) [22]) with the trend, seasonal and irregular components.

In the next subsection we will review a more common for IoT system case of time series - the multivariate ones.

2.1.2 Multivariate Time Series Models

The univariate time series in IoT systems are best represented by independent variables, which must be considered individually. For example, air temperature on the street of the town. However, typically, the IoT sensors are subjected to the same influences simultaneously, which create ground for interdependencies between the generated data streams. Such interdependencies manifest themselves as correlations between the variables. For example, engine's oil temperature may depend on the vehicle acceleration and speed. In such cases the interdependence of the data streams may be better revealed by combining such streams into joint multivariate time series. If the interdependence can be justified by

known physical reasons, then unification of the interrelated time series would be justified by the contextual basis available for those reasons. The Kalman filter algorithm is capable of processing either type of time series as well as mixed varieties. This includes univariate and multivariate time series, as well as independent or interdependent ones.

In each case, the design of the Kalman filter would reflect the type of time series involved via selection of state variables as well as the structure of the state transition matrix, which formally summarizes the inter-relations between the variables involved. In turn, the correlations between the time series are reflected via non-zero non-diagonal elements of the covariance matrix.

Essentially, due to this ability to reflect the existing inter-dependencies between the time series, multivariate analysis presents a more sophisticated and accurate reflection of IoT related systems. In a way, it overcomes the narrow limitations of individual data streams to capture the dynamic nature of IoT systems.

Generalization of univariate time series to a multivariate time series version is achievable via substitution of the variable y_t as in equation 2.3 to a column vector of N observable variables $y_t = (y_{1,t}, \ldots, y_{N,t})^T$. Hidden Markov chain model still stands for multivariate time series case, for which the notations of Figure 2.1 include also state vector $s_t = (s_{1,t}, \ldots, s_{N,t})^T$, and state transition matrix $\mathbf{A}$ as well as observation matrix $\mathbf{C}$. In univariate case A and C were just transform coefficients.

For the local level model the multivariate version of it becomes

$$y_{i,t} = \mu_{i,t} + \varepsilon_{i,t} \tag{2.5}$$

with

$$\mu_{i,t} = \mu_{i,t-1} + \eta_{i,t} \tag{2.6}$$

for all $i = 1, \ldots, N$ and $t = 1, \ldots, T$.

For the IoT environment multivariate models are particularly attractive due to their ability to rely on versatile specifications for all related components, including trend, cyclical and seasonal components. For example, the existence of common trend is suggestive of co-integration of variables at a zero frequency. Similarly, the presence of common cyclical and seasonal components in the time series is a result of co-integration of variables at higher frequencies, suggesting the existence of a common source for cyclical and seasonal patterns in the related time series. For the fleet management applications, cyclical component may be illustrated as day/night sequential changes of vehicle operation. Seasonal patterns of engine's oil viscosity at spring, summer, autumn and winter temperature conditions illustrate effect of the weather on related time series changes.

2.2 Alternatives to Structural Time Series

The viable alternative to structural time series modelling is a technique called an autoregressive integrated moving average (ARIMA). There are two areas in which the ARIMA approach and structural time series overlap: on one hand, ARIMA does allow state space representation (examples are available from Harvey(1989) [13], and Brockwell and Davis (1991) [3]), while on the other hand, some structural time series do accept ARIMA specifications (see Hamilton (1994) [12]).

In comparison to structural time series, the ARIMA procedure is less transparent, as it applies differencing to initial time series in order to exclude trend and seasonal components prior to the implementation of the analysis. Conversely, in structural time series all the components, including trend, cyclic and seasonal, are expressed explicitly, thus offering full transparency in terms of any changes that occur. As a result, this allows special treatments to be employed in particular circumstances of potential importance. Thus, the structural time series approach offers more flexibility and transparency for processing. This may be advantageous when there is a need to verify whether predictions are in line with expectations, based on available data. Additionally, ARIMA differencing, which excludes trend and seasonal components upfront, may be preferred when the task requires that conclusions are not affected by the long-term trend and periodic seasonal variations.

Moreover, having structural series at hand, as Moritz et al (2015) [22] assert, it is possible to decompose this series and have explicit presentation of the components, as it is shown in Figure 2.3. From this viewpoint, the reverse operation would allow us to artificially generate time series of the predefined types, which forms the basis for synthetic data generation with desirable properties.

On the preferrence issue between the alternatives, there is indication from Harvey and Todd (1983) [14] favouring the structural time series approach over ARIMA in forecasting procedures for practical tasks.

Advantages of the Kalman filter algorithm which naturally relies on state space representation and operates recursively are emphasised by Brockwell and Davis (1991) [3] in relation to the analysis of time series with missing observation data. This conclusion is based on evaluations of the Gaussian likelihood function at the observed data points, while maintaining minimum mean squared error for missing data points. A similar conclusion is reached by Jones (1993) [19] based on employment of the state space approach to longitudinal data with serial correlation.

Processing data streams with missing data points is more time consuming with ARIMA procedures as compared to that based on state space equations. Moreover, models based on state space representation allow the direct conversion of univariate structural time series into multivariate ones, which is not possible within ARIMA routines. Additionally, the recursive nature of state space algorithms, such as the Kalman filter, ensures processing of the multivariate time series with limited increase of computational complexity, which has practical importance.

When some data are missing, the ARIMA method allows imputation of missing data according to the statistical model of the available points in time series. In the recent

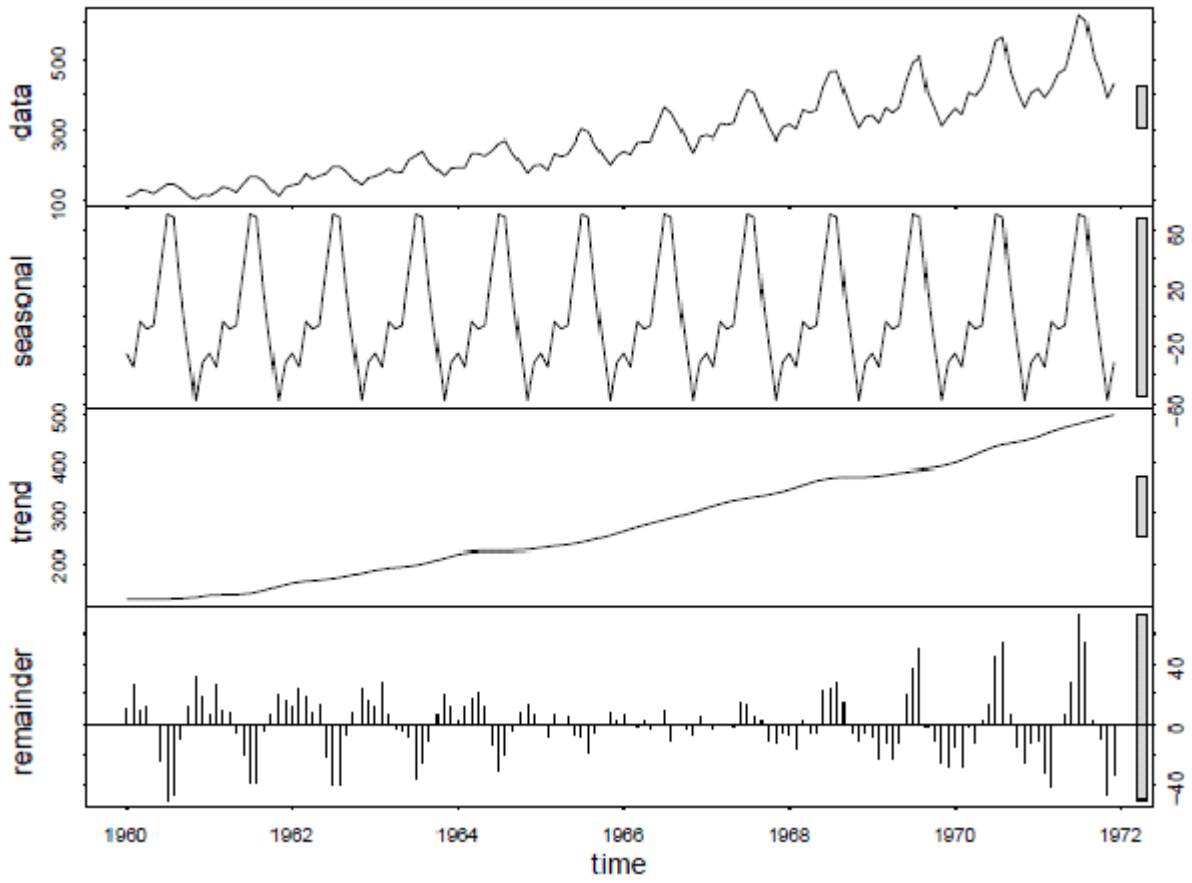
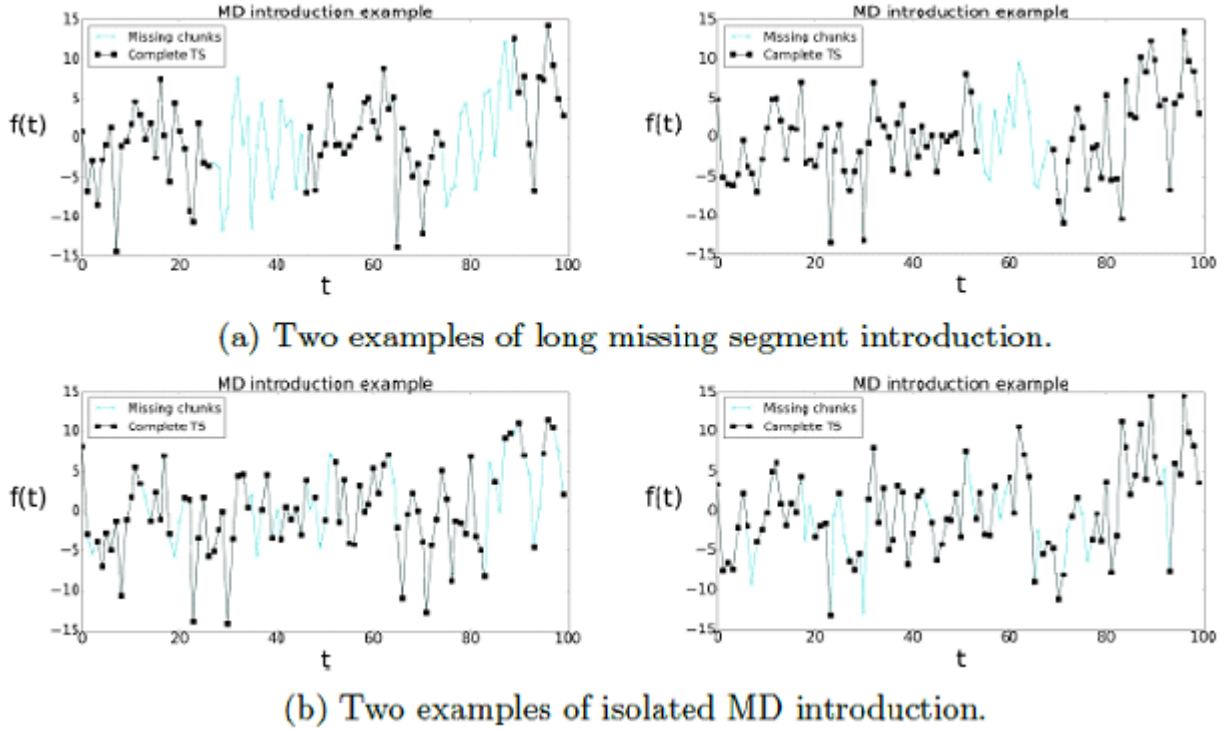


Figure 2.3: Seasonal decomposition of airpass TS dataset (according to Moritz et al (2015) [22])

research work, [10], the author, Garciarena (2016), has compared four basic imputation methods of missing data: (1) interpolation, (2) Kalman-based, (3) multiple imputation using chained equations (MICE) and (4) regression. Examples of missing data (MD) in time series (TS) used in [10] are shown in Figure 2.4. For multiple imputation using chained equations (MICE) as a sub-method, this work used a boot-strap algorithm implemented in the R package MICE [4]. An important conclusion of this work is that in terms of time consumption, the biggest difference was found between Kalman filter model imputation and MICE. MICE almost doubled the time growth rate of the Kalman filter. Overall, it was determined that MICE based regression is the best imputation method out of all four methods considered. If there is no need for fast imputation, its usage is absolutely recommended. It should be noted here that for IoT systems this would more likely correspond to conditions with the stored data, rather than real time data streams. However, in cases that require a considerably fast imputation, such as real time data, Kalman-based regression could be the superior choice, and could be considered the top imputation method. In sum, according to Garciarena (2016), [10], using MICE as a basis for subsequently performing regression produced better results than the Kalman filter algorithm in terms of imputation

value accuracy. However, the Kalman filter offered much better results when computational time consumption was the main concern. From this we may deduce that for IoT system in-situ data stream pre-processing, Kalman filter should be the algorithm of choice to manage the initial data flows from the sensor networks of IoT systems.



(a) Two examples of long missing segment introduction.

(b) Two examples of isolated MD introduction.

Figure 2.4: Four examples of missing data (MD) introduction in TSs (from Garciarena (2016) [10]).

2.2.1 Simulation of Missing Data

In the research area of missing data imputation there exists the issue of the data itself. This dilemma has been addressed by Moritz et al (2015) [22] who formulated the grounds for the simulation of missing data. The authors indicated that evaluating the performance of imputation algorithms has one intrinsic difficulty, namely, that comparing the results of truly missing data is not possible, since the actual values are in fact missing and therefore inaccessible. Thus it will never emerge how much the imputed values differ from the real values. A performance comparison can only be done for simulated missing data, i.e. when a complete series is taken and data points are artificially removed. Once the imputation algorithm is applied to the remaining data and datasets for imputation are produced, then the imputed values and the previously removed real values can be compared to verify the validity of the applied imputation procedure. This concept justifies the use of synthetic data for the research work of this thesis in its imputation part.

2.3 The Kalman Filter Operation

The Kalman filter algorithm represents an important procedure in estimating dynamically changing systems in its state space format based on observable non-state parameters. In this section, an introduction to the Kalman filter algorithm is outlined. The intent is to demonstrate the connection between Kalman filter, state space and IoT parameters.

2.3.1 Recursions in the Kalman Filter Algorithm

Historically, the goal of minimizing statistical errors when fitting noisy data led to the concept of least squares regression, which was initially introduced by Legendre (1805). This theory was first published in 1805 and named by Legendre "least squares". Appreciation of this theory was dated four years later, when Gauss (1809) developed the method as a statistical routine for data processing and incorporated the least squares technique into the procedure to estimate the error terms via a probabilistic approach (D.S.G. Pollock,(2003)) [24].

The Kalman filter delivers optimal solutions in terms of minimizing the mean of the squared error, which is achieved via the recursive procedure of first estimating and then updating the estimation of the system's state in time $t+1$ based on available state estimation at a previous time slot, t . Thus, the Kalman filter computes a linear, unbiased and optimal estimation of the recursive solution according to the least squares approach. As such, the Kalman filter algorithm consists of a set of equations governing the transition of the system from the state at time t to the state at time $t+1$.

An essential step in enabling operation of the Kalman filter is proper selection of the state variables. The set of selected state variables aims to reflect the condition of the system in the most concise and relevant manner within the context of the performed task. The number of the state variables should be minimized to reduce computational complexity and improve convergence of the recursive iterations. However such reductions should not exclude relations inside the system, which may prove to be essential at the end.

The operation of the filter assumes that the system is linear in a sense that its behaviour is described by a set of linear stochastic difference equations, allowing estimation of the state $\mathbf{S}$ of the system based on the state transition equation

$$\mathbf{S}(k+1) = \mathbf{A} \cdot \mathbf{S}(k) + \mathbf{B} \cdot \mathbf{u}(k) + \eta(k) \tag{2.7}$$

and the measurements equation

$$\mathbf{Z}(k+1) = \mathbf{C} \cdot \mathbf{S}(k+1) + \varepsilon(k+1), \tag{2.8}$$

where $\mathbf{S}$ is a state vector comprised of a set of n state variables, $\mathbf{A}$ is a state transition matrix governing transformation of the system's state from discrete time moment number k, i.e. t_k, to the next discrete time moment number $k+1$, i.e. t_{k+1}, with $k = 0, 1, 2, 3, \cdots$.

Having n state variables, the dimensionality of matrix $\mathbf{A}$ is $n \times n$. The matrix $\mathbf{A}$ may also be time-dependent, in which case the enumeration index would apply. However, within the scope of this study we limit consideration by stationary case for matrix $\mathbf{A}$, i.e. when $\mathbf{A}$ does not change with time. Although a simplification, it nevertheless is valid for a large number of cases and is regarded to be of practical importance. Vector $\mathbf{Z}$ is a set of m observables, which are m parameters subjected to monitoring via sensor networks. The $n \times m$ observation matrix $\mathbf{C}$ performs transformation from state space to sensors space, connecting the state vector $\mathbf{S}$ with the observation vector $\mathbf{Z}$.

The term $\mathbf{B} \cdot \mathbf{u}(k)$ in equation 2.7 relates to the optional control unit for the system, where vector $\mathbf{u}$ holds l control variables, whose influence on the system's state is determined by $n \times l$ matrix B.

The vector terms ε and η are independent random noise components, which are Gaussian distributed (1) process noise ε for the state variables and (2) measurement noise η for the sensors:

$$p(\,\varepsilon\,) \sim N(0,\, \mathbf{Q}) \tag{2.9}$$

$$p(\,\eta\,) \sim N(0,\, \mathbf{R}) \tag{2.10}$$

Here $\mathbf{Q}$ is the $n \times n$ process covariance matrix, while $\mathbf{R}$ is the $m \times m$ measurement covariance matrix, both of which are assumed to be known for their use in the Kalman filter. Either of the covariance matrices, $\mathbf{Q}$ and/or $\mathbf{R}$, may have time dependences, however within this study we restrict it to stationary behaviour.

The iterative cycle of the Kalman filter consists of two stages - the stage projecting the state of the system one step ahead in time based on the current state estimation and the state transition equation 2.7, which is the "(I) Predict stage", followed by the "(II) Update stage". The last one compares the prediction made in the first stage with the arriving new set of noisy measurements to draw conclusions about the actual achieved state as follows.

I. Predict stage:

Step 1. Predict the state $\widehat{\mathbf{S}}$ one step ahead:

$$\widehat{\mathbf{S}}(k+1) \leftarrow \mathbf{A} \cdot \widehat{\mathbf{S}}(k) + \mathbf{B} \cdot \mathbf{u}(k) \tag{2.11}$$

Step 2. Predict the error covariance $\mathbf{P}$ one step ahead:

$$\mathbf{P}(k+1) \leftarrow \mathbf{A} \cdot \mathbf{P}(k) \cdot \mathbf{A}^{\mathbf{T}} + \mathbf{Q} \tag{2.12}$$

II. Update stage:

Step 1. Compute the Kalman gain $\mathbf{G}$ coefficient:

$$\mathbf{G}(k+1) \leftarrow \mathbf{P}(k+1) \cdot \mathbf{C}^T / \{\mathbf{C} \cdot \mathbf{P}(k+1) \cdot \mathbf{C}^T + \mathbf{R}\} \tag{2.13}$$

Step 2. Update the state estimate $\widehat{\mathbf{S}}$ with the measurement $\mathbf{Z}$

$$\widehat{\mathbf{S}}(k+1) \leftarrow \widehat{\mathbf{S}}(k+1) + \mathbf{G}(k+1) \cdot [\,\mathbf{Z}(k+1) - \mathbf{C} \cdot \widehat{\mathbf{S}}(k+1)\,] \tag{2.14}$$

Step 3. Update the error covariance $\mathbf{P}$

$$\mathbf{P}(k+1) \leftarrow [\,\mathbf{I} - \mathbf{G}(k+1) \cdot \mathbf{C}\,] \cdot \mathbf{P}(k+1), \tag{2.15}$$

where $\mathbf{I}$ is $n \times n$ identity matrix, $\mathbf{P}$ is $n \times n$ error covariance matrix, and $\mathbf{G}$ is $n \times m$ Kalman gain coefficient matrix. At the starting point of the Kalman filter algorithm the initial condition apply at $t = t_0$, reversing equation 2.8

$$\mathbf{S}(0) = \mathbf{C}^{-\mathbf{1}} \cdot \mathbf{Z}(0) \tag{2.16}$$

The solution is optimal provided the filter combines all observed information and previous knowledge about the system's behaviour such that the state estimation minimizes the statistical error. The recursive term means that the filter re-computes the solution each time a new observation is incorporated into the system.

The Kalman filter is the main algorithm for estimating dynamic systems in state space form. This representation of the system is described by a set of state variables. The state contains all information relative to that system at a given point in time. This information should allow the inference of past behaviour of the system, aiming to predict its future behaviour. What makes the Kalman filter so interesting is its ability to predict the state of a system, even when the precise nature of the modelled system is unknown. In practical terms, the individual state variables of a dynamic system cannot be exactly determined through a direct measurement. In this context, their measurement is done by means of stochastic processes involving some degree of uncertainty.

2.4 Pros and Cons of the Kalman Filter for IoT

2.4.1 Advantages

For the IoT field, a Kalman filter algorithm possesses a number of useful features, making it an attractive option to pre-process of the raw data streams on-the-fly. These advantages are outlined below.

1. Fast estimates. This feature comes from the fact that the Kalman filter is producing its estimate in a single iteration step. It relies on the current state and makes predictions via state equations, followed by estimation update on arrival of the measurements. This is essential for on-the-fly pre-processing because of the time constraint imposed by the often sufficiently high speed of the processes involved.

2. Adaptability to structural breaks. The unexpected changes in time series, such as change of the mean or variance, which pose a challenge to regression techniques as a change in the regression model, does not necessarily cause failure of the Kalman filter. Rather, it results in additional re-training and adjustment of the Kalman filter to new conditions. Thus, the Kalman filter recursively estimates the stochastic path of the coefficients in the model, allowing changes to occur and compensating it via training to convergence in the changed models. For IoT systems, this means that the Kalman filter offers flexibility in adjusting to model changes.

3. Minimized error. The Kalman filter estimates the state vector by minimizing the variances, as it relies on the least squares method to recursively derive its estimations. For the IoT systems it brings in an optimized accuracy of the estimations made.

4. Wide range of models. The Kalman filter offers dynamic modelling of linear systems, which covers many models from the IoT area, where individual "things"behave according to such models. Linear transitions from one time moment to the next one are a very common transformation of "things"in IoT systems. Therefore, such description by Kalman filters is beneficial for the IoT field.

2.4.2 Disadvantages

There are, however, some shortcomings which require extra consideration when involving Kalman filters to describe systems in IoT. The following are features, which may be considered as disadvantages when employing the Kalman filter algorithm.

1. The need to set up the initial conditions. To implement the Kalman filter, the initial means and the variances of the state variables must be specified. Availability of the training set of data, on which the Kalman filter converges, presents the solution to this problem and thus limits its negative effect on the usability of the Kalman filter algorithm in IoT.

2. Limitation to Gaussian distributions. The Kalman filter requires noise components to have Gaussian distributions. This restriction still accommodates a wide range of practical cases from IoT field.

3. Sensitivity to correctness of the state equations. The state equations empower the Kalman filter when they correctly reflect the behaviour of the system, allowing the filter to pick up the underlying trends and converge quickly. However, if the system's actual behaviour differs significantly from that described by the state equations, then convergence of the filter suffers, and so does its usability in IoT. Therefore, the selection of the state variables as well as state equations becomes crucial for the properties of the resulting Kalman filter and its usefulness to IoT applications.

Overall, the advantages of the Kalman filter makes it a preferred tool for pre-processing IoT data streams under strong time constraints, particularly when possible disadvantages are efficiently addressed.

2.5 Multi-step Prediction and the Kalman Filter

Terminologically, the term "prediction", used towards Kalman filter output from the first half of its cycle when the current state is converted to the next time slot state via application of the state equations, is not strict. The same is true with regard to the repeated application of this half-cycle under circumstances of missing measurements, which would result in multi-step "prediction". Despite the fact that state equations are formulated and applied here by the Kalman filter, its output does not fit the definition of actual prediction as it is outlined by prediction theory (see, for example, [1], [11], and particularly [31]).

The actual single- or multi-step prediction becomes an extension of control theory, which uses difference or differential equations to create mathematical models of the designed systems. The examples are missile guidance systems using single-step prediction functions with the smallest relevant time constant $\delta\tau$. The following classification of functionalities are defined in the literature:
(i) smoothing — it includes functions operating at $\delta\tau < 0$;
(ii) filtering — it includes functions operating at $\delta\tau = 0$;
(iii) prediction — it includes functions operating at $\delta\tau > 0$.
Within such classification criteria the single-step prediction cycle of the Kalman filter, [20], qualifies as "filtering", not as "prediction". These terminological differences between "filtering"(as estimation) and "prediction"have been rectified by Athens and Kendrick, [2], according to whom the multi-step prediction at time slot of the future requires ability to measure the accuracy of such prediction and this in turn implies the following:
(i) prediction must be defined in terms of a specifications of possible outcomes;
(ii) the probability that the prediction will occur must be given and supported by the history of prior predictions;
(iii) the probability that the prediction will occur in the prediction statement must be based on data that do not take part in producing the current prediction;
(iv) the probability that the prediction will occur must be accompanied by a confidence level.

An example of a Prediction System is given in Figure 2.5 for predicting future outcomes from the systems with observable outputs, see [7]. This design takes the observable factors as inputs for the prediction system, thus making them a basis for prediction. On the other hand, the unobservable factors are used statistically by an estimation subsystem, such as Kalman filter, to form a basis for the future estimation or forecast.

Here a set of errors is formed, calculated as the difference between predictions and the actual system outcomes within the available past history or a looking back horizon. This set of errors is then used as a measure of the prediction accuracy. The data on the looking back horizon must be statistically sufficient to deduce the confidence interval for the error probability statement. If the last requirement is satisfied, then a qualified prediction can be made with appropriate accuracy. Otherwise the produced outputs qualify as forecasts, which is still sufficient for the Kalman filter applicability essential in IoT systems.

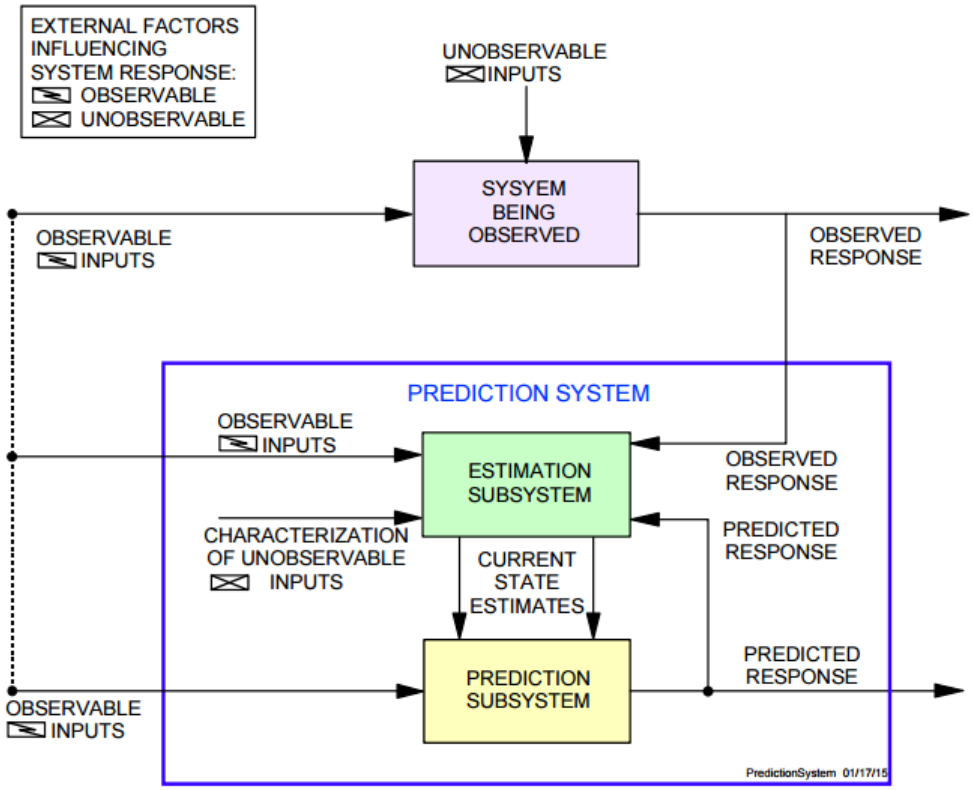


Figure 2.5: Schematic of a Prediction System (reproduced from [7]).

Chapter 3

Proposed Architectures

3.1 Defining the Compositional Kalman Filters

Sophisticated objects of IoT systems, such as smart vehicles, may rely on sufficiently large number of sensors dedicated to monitor parameters of its various parts and modules, which, in turn, characterize state of the related objects. As the number of the parts grows with the complexity of the IoT objects, so does the number of state variables to fully characterize it. A straightforward approach to build a Kalman filter to monitor state changes of such object would be to design a single filter with a full set of state variables. In case of sufficiently large number of state variables, the problems of computational complexity as well as filter convergence would arise, rendering such solution to be impractical. The alternative to the above straightforward approach would be to separate various data streams between specialized filters with the minimal number of state variables each. This approach, however, has its limitation imposed by relevant state equations. For instance, because the equation of motion does include acceleration, excluding acceleration related state variables from the definition of respective Kalman filter would be counter-productive due to reduction of accuracy of such Kalman filter estimates and additionally jeopardizing its convergence. Optimal solution in this case would be seen in grouping the available data streams on IoT's contextual basis and relating them to specialized Kalman filters. These specialized Kalman filters, serving the mini-groups of IoT system's data streams, we call elementary Kalman filters of IoT object. Together, as a set, these elementary Kalman filters would play the role of monitoring the state of the IoT object as a whole, thus being equivalent to the initially discussed single filter with a full set of state variables but with reduced computational complexity each and reduced problems with convergence. This would address the task of monitoring state of the IoT object. The procedure of unifying the elementary Kalman filters to represent state of the whole object may be called Kalman filter integration. The designed architecture of the elementary Kalman filters for large system representation we may call integrated Kalman filter.

However, among those elementary filters there will be some with overlapping sets of state variables, which therefore will be estimating the same state variables from different perspective. This, in turn, does create certain redundancy in the estimates of the state

variables, as different estimates of the same variables become available from different elementary Kalman filters. For such occasions we have to derive the best estimate from the set of estimates given by different filters for the same variable. This can be done by some specific algorithm, which can be implemented via architectural interfacing of related elementary Kalman filters. Such architectural composition of the connected Kalman filters, in which various versions of state variable estimates are algorithmically processed to produce the best single estimate of that variable to be accepted for all relevant Kalman filters, we call compositional Kalman filter. In this definition we have the following counterparts constituting the compositional Kalman filter: 1) elementary Kalman filters with overlapping sets of the output state variables; 2) the connecting interface, which unites all outputs to algorithmically produce a single output for the related architecture.

As an example, let us consider Kalman filters related to the navigational section of the smart car. The equation of motion includes such variables as position, velocity and acceleration for each coordinate axis. Therefore in two-dimensional space there will be 6 state variables describing the motion, which we can choose to be a set $[x, y, v_x, v_y, a_x, a_y]$. The Kalman filters, which can contain this set of state variables include the following: GPS-KF, V-KF, A-KF, GPS/V-KF, GPS/A-KF, GPS/V/A-KF. Here we adopt the following notations: KF stands for Kalman filter; GPS-KF denotes Kalman filter relying on GPS signal only; V-KF is KF relying on velocity components only, i.e. v_x and v_y; A-KF is KF relying only on acceleration components, i.e. a_x and a_y; respectively mixed types of sensory foundation of KF is denoted by listing all the sources via slash, such as GPS/V-KF when both, GPS and V-components, support KF, GPS/A-KF for GPS and A-components as the sources, and, finally, GPS/V/A-KF for KF with all the above sources used). These are elementary Kalman filters with the set of state variables being $[x, y, v_x, v_y, a_x, a_y]$ for each filter. Because all of these filters rely on the same set of state variables, which is the partial case of overlapping sets of state variable when the sets are identical, the various pairs of such elementary Kalman filters may constitute the compositional units of Kalman filters, as it is illustrated by Table 3.1, where listed are some of the possible pairs of the elementary Kalman filters and given are the related input variables (more specifically, the input variables converted to its state variables counterparts), as well as output variables (these are represented by selected set of state variables). Table 3.1 may be extended further to include more possible combinations.

Furthermore, Table 3.2 presents the logistic of building up compositional architectures from some of the pairs taken from Table 3.1. It is worth noting, that the same pairs of elementary Kalman filters may produce different compositional architecture due to variations in interconnecting interface, which specifies algorithmic processing of the outputs from the constituents. For example, the pair GPS/V-KF and V-KF may be connected via interfaces J_1, J_2 or J_3, producing different compositional architecture: GPS pinning architecture in case of J_1 interface, switching architecture to handle GPS outage in case of J_2 interface, and GPS offset detection/compensation architecture when employing interface J_3 (see Table 3.2).

The scope of this thesis work is limited to the case of the very first line of the Table 3.1, which was expanded to three different cases listed in the Table 3.2 (see first three lines of the Table 3.2). Multiple other cases mentioned in the Table 3.1 and Table 3.2 are to

illustrate the existing perspective and the scope of possible research in the future related to the introduced notion of compositional Kalman filters. Next, we shall consider in more details those selected cases from Table 3.1 and Table 3.2, that adduced the scope of the present thesis.

№	Kalman filter pairs	Input variables (converted to its state counterparts)	Output state variables
(i)	GPS/V-KF; V-KF;	x, y, v_x, v_y;	x, y, v_x, v_y, a_x, a_y;
(ii)	GPS/A-KF; A-KF;	x, y, a_x, a_y;	x, y, v_x, v_y, a_x, a_y;
(iii)	GPS-KF; V-KF;	x, y, v_x, v_y;	x, y, v_x, v_y, a_x, a_y;
(iv)	GPS-KF; A-KF;	x, y, a_x, a_y;	x, y, v_x, v_y, a_x, a_y;
(v)	V/A-KF; A-KF;	v_x, v_y, a_x, a_y;	x, y, v_x, v_y, a_x, a_y;
(vi)	V-KF; A-KF;	v_x, v_y, a_x, a_y;	x, y, v_x, v_y, a_x, a_y;
(vii)	GPS/A-KF; V-KF;	x, y, v_x, v_y, a_x, a_y;	x, y, v_x, v_y, a_x, a_y;
(viii)	GPS-KF; A-KF;	x, y, v_x, v_y, a_x, a_y;	x, y, v_x, v_y, a_x, a_y;

Table 3.1: Combination examples of the pairs of elementary Kalman filters which may serve to build compositional units via interconnecting them with interfaces.

№	Interface	Compositional pairs	Interface functionality
(i)	J_1	GPS/V-KF; V-KF;	Eliminate integration constant of V-KF via GPS pinning procedure
(ii)	J_2	GPS/V-KF; V-KF;	Switch from GPS-based monitoring of motion to GPS-free one, providing smooth transition for handling GPS outage event
(iii)	J_3	GPS/V-KF; V-KF;	Detect the event of GPS offset occurrence and compensate it
(iv)	J_4	V-KF; A-KF;	Detect the events of wheels slipping on the roads surface
(v)	J_5	GPS/V-KF; A-KF;	Detect the events of wheels slipping on the roads surface and correct the distance measurement
(vi)	J_6	V-KF; A-KF;	Detect the events of emergency breaking
(vii)	J_7	V-KF; A-KF;	Detect the events of the car sliding to spin around the axis, which is perpendicular to the road's surface

Table 3.2: Description of selected compositional Kalman filters

3.2 Outline of Proposed Architectures

In the following research, we propose the building of architectures which include combinations of Kalman filters (KFs) to efficiently address data stream processing in IoT systems. Such architectures are composed from elementary KFs as building blocks, connected via interfaces to implement the logistics of the operations relevant to IoT systems. Such compositional designs aim to either improve the efficiency and performance of individual KFs or go beyond the capabilities of individual KFs and offer extra functionalities, that are only achievable by inter-linked groups of KFs. For that reason, we call these architectural designs compositional Kalman filters.

The compositional architectures herewith aim to advance the performance of basic operations over data streams in IoT environments, such as
(a) imputation of intermittent data,
(b) recovering of lost data and/or replacing missing data,
(c) detecting, eliminating and/or correcting errors.

A specific resource that is targeted here is the redundancy in data streams. To take advantage of this redundancy in terms of improved performance of known operations or new functionalities not available otherwise, such as special events detection, we utilize the interconnections between the KFs. To ensure interaction between the KFs we design interfaces between them, that compare their predictions and algorithmically resolve differences when detecting logically interpretable relationships.

To explore the subject, we have chosen the case of navigational data streams of the travelling vehicle. This choice is motivated by the fact that the basics of the vehicle navigation via KF are very well studied for the purposes of navigation itself, i.e. for control purposes, but not so for the purpose of events detection in the IoT environment. Here, the emphasis is on the observation of the data and establishing relations essential in detecting navigation-related events. Specific usage of the detected events is not necessarily for control purposes, but rather to achieve the above goals relevant to IoT systems, such as detecting and eliminating errors, preventing data loss, and replacing missing data. Table 3.3 provides an outline of case-specific scenarios that we have used here to address the subject from various perspectives.

№	TARGETED OPERATIONS	IMPLEMENTATIONS
(i)	removing Gaussian noise	cleaning noise from the data
(ii)	intermittent data imputation	GPS data frequency upgrade (1 Hz to 10 Hz conversion)
(iii)	missing data recovery	GPS outage event
(iv)	outliers/errors detection and correction	GPS offset: events of global and local offset detection and correction

Table 3.3: List of IoT targeted operations and simulation scenarios for their implementations covered herewith.

3.3 Data Streams in Proposed Architectures

The data streams in IoT systems originate from sensor networks, which monitor various parameters of the constituent devices of such systems within the time flow. These initial generated data are first stored at the local level, where they are pre-processed and prepared for transfer to the cloud-based server. Such transfer occurs when the connection to the cloud becomes available, and as a result the data undergo further processing and eventual placement in the database in a structured format, allowing deeper analytical review. Reviews of this nature may be conducted by way of analytical queries, which aim to establish the presence or absence of certain correlations within the data. Thus, data streams of IoTs are operating on at least the following three levels:
1) local data streams from sensors to local data storage systems;
2) data streams from local storage to the server;
3) data streams generated by a database to answer queries from users on the retrieval of specific data sets.

These levels are sequentially connected in terms of converting the initial raw data from the sensors into structurally organized data sets. However, the operation of each level is organized according to its own logistics, as illustrated in Figure 3.1 through Figure 3.3.

Specifically, Figure 3.1 illustrates the local data flow level, where all data streams from the sensors network undergo on-the-fly pre-processing prior to local storage, passing through the set of KFs designed to accommodate specific groups of streams in order to reduce noise, eliminate outliers, fill in lost or missing data, correct errors, and compensate for potential data distortions. Because of its fast performance, the KF becomes a front runner as the algorithm of choice for the on-the-fly data pre-processing. Moreover, in this research we propose that the pre-processing unit is to be composed of sets of specialized KFs, which are designed to account for redundancies in the existing data streams in order to efficiently detect erroneous data, fix detected errors, as well as compensate for lost or missing data. In this regard, data streams are to be grouped contextually to accommodate the existing relations between data and take advantage of redundancies. We propose calling such pre-processing units composed of specialized KFs a compositional KF to indicate that its design reflects the existing interdependencies in data streams and aims to account for those in the data pre-processing phase. The focus of this research effort is to demonstrate principles related to the operation of compositional KF architectures as exemplified by processing of the navigational data streams of vehicles.

An on-the-fly pre-processing unit allows for initial noise reduction in data streams as well as the correction of some errors. It also compensates for missing data, after which data are stored locally and thus become available for further processing. Stored data accommodate a wider variety of processing algorithms compared to the real-time data streams. This is shown in Figure 3.2 as exchange between the storage unit and the first pre-processing unit. To exemplify those additional algorithms for the stored data, in Figure 3.2 in addition to the KF also shown are backward KF (abbreviated BKF) as well as other statistical procedures unified under the abbreviation OT. These additional algorithms have the potential to further improve the signal-to-noise ratio in the data as well as the accuracy of error corrections and adjustments to statistical properties of imputed data . Final data

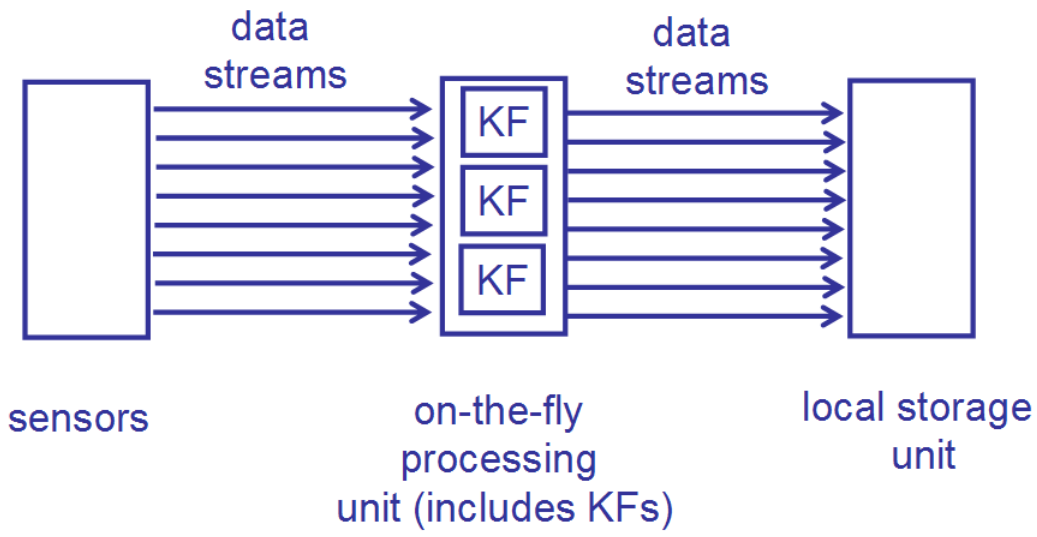


Figure 3.1: Local data streams from sensors to local data storage systems.

verification as well as compression prior to cloud server transfer may be performed by the optional second pre-processor shown in Figure 3.2, if necessary.

Finally, when on the cloud, the data will be placed in the database to allow retrieval in various formats in response to analytical queries, as illustrated by Figure 3.3. Here the user-initiated queries will be converted into data requests of specific formats. The retrieved raw data may then undergo specialized KF processing (for example, equalizing the data frequencies if they differ in the raw data streams) to form the data stream responses. Such possibilities present implementation of the eventual data storage configuration of IoT systems, which goes beyond the scope of this research. However, it should be noted as a potential area for future expansion. This current work focuses on its first level, namely on-the-fly pre-processing of raw data streams as illustrated in Figure 3.1.

3.3.1 Architectural Assumptions and Restrictions

There are a number of assumptions and restrictions which are explicitly and/or implicitly used in the proposed architectures that limit their applicability.

Firstly, the architectures here are assumed to operate with two-dimensional navigational data streams , in which the vehicle orientation angle θ is given as a road design parameter and can thus be retrieved visually. For example, this can be done with camera sensors, which verify how accurately the driver is following the road track, or alternatively with a gyroscope for the same purpose. Note here that deriving θ from the GPS signal only as $\theta = \arctan((dx/dt)/(dy/dt))$ would deprive the system of the θ values during GPS outage events. Consequently, it is advantageous of having θ value source to be independent of the GPS signal.

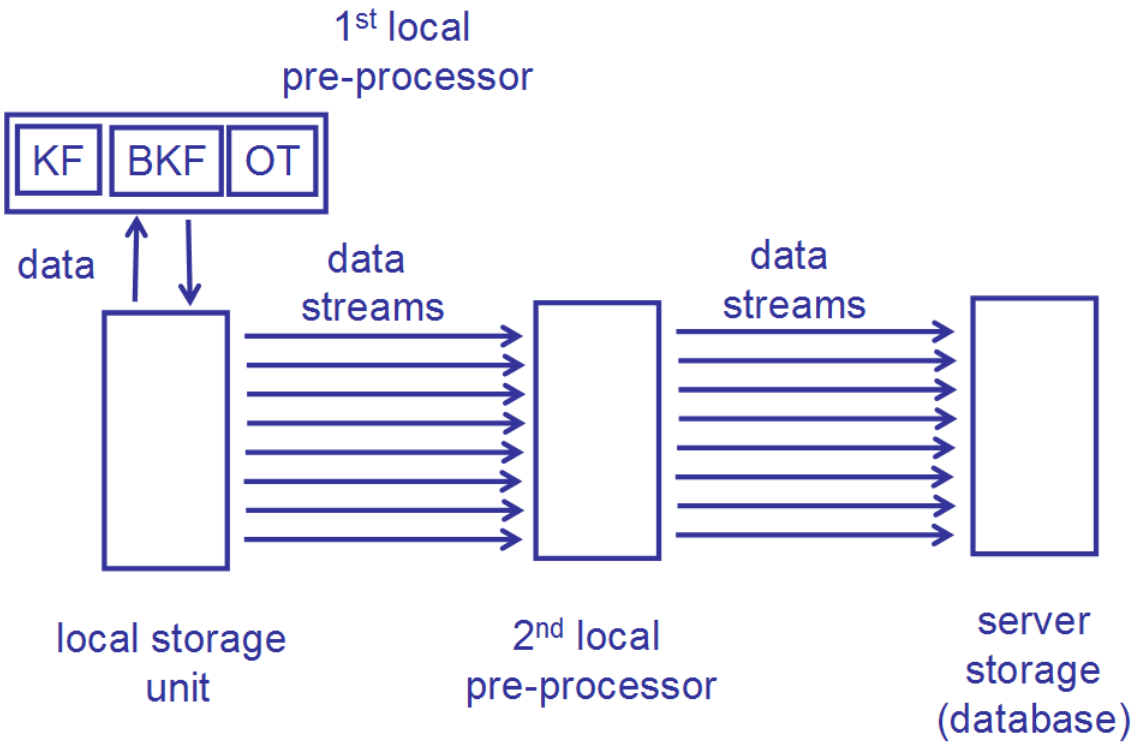


Figure 3.2: Data streams from local storage to the server.

For our architectures there are two sources of noise which may affect the GPS signal values, both of which must be accounted for in the implementation of the Kalman filter algorithm as well as its simulation and performance analysis, presented herewith in Chapter 4. The process noise is characterized by covariance matrix Q and within this study it is chosen to be sufficiently small so as to have no practical effect on the outcome of the experiments. Specifically, for the GPS signal, the diagonal elements of covariance matrix Q contributing to values of x and y on the process level have been set to be $Q_{11} = Q_{22} = 10^{-9}$ m^2. The measurement noise is characterized by covariance matrix $\mathbf{R}$, which is a square matrix with its size defined by the number of sensors in the model, with its diagonal elements representing variances of the respective sensors. The maximum number of sensors considered herewith is 6, including 2 GPS sensors for latitude x and longitude y, 2 sensors for velocity components v_x and v_y (originating from speedometer and orientation meter merged to produce v_x and v_y as in Figure 3.4), and 2 sensors for accelerometer components a_x and a_y (built as merged accelerometer and orientation meter, similar to velocity components). The largest size of the $\mathbf{R}$ matrix herewith is therefore 6-by-6, with R_{11} and R_{22} being variances of x and y, R_{33} and R_{44} - variances of v_x and v_y, R_{55} and R_{66} - variances of a_x and a_y. We assume that there is no cross-correlation between the sensors, and therefore all non-diagonal elements of $\mathbf{R}$ are zeros, i.e. $R_{ij} = 0$ for $i \neq j$. In addition to this numerical indexing by position in the matrix (R_{ij} with i for row number and j for column number) we use indexing by variable name, so that the above R_{11} is also R_x, R_{22} is R_y,

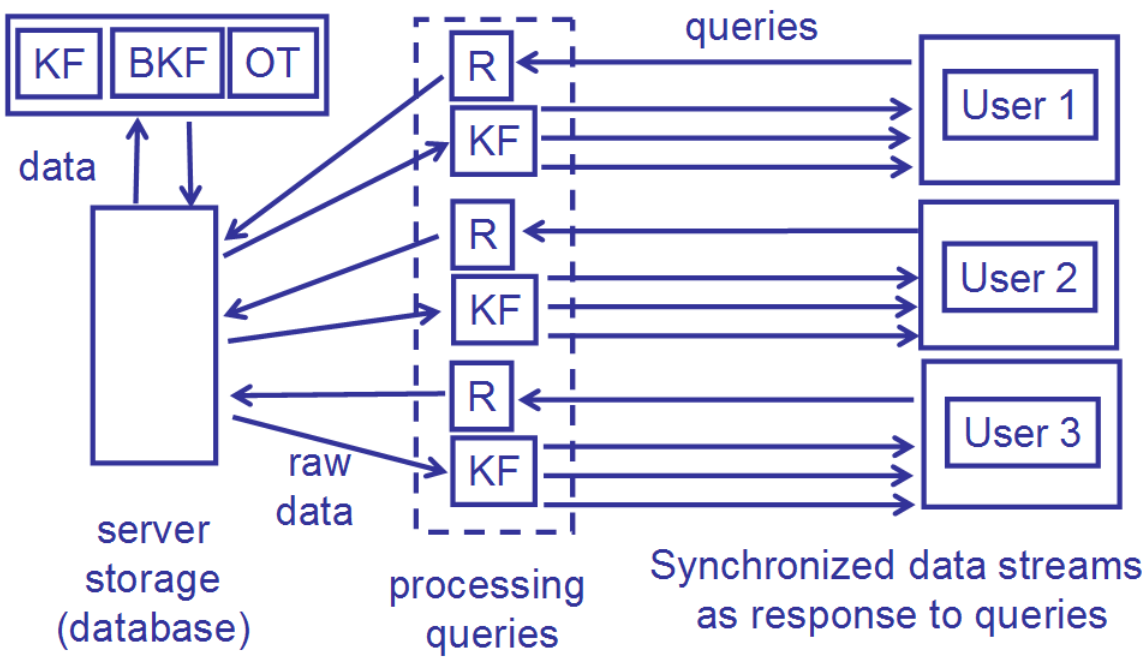


Figure 3.3: Data streams generated by database to answer the queries from users on retrieval of specific sets of data.

R_{33} is R_{v_x}, R_{44} is R_{v_y}, R_{55} is R_{a_x}, and R_{66} is R_{a_y}. Two different indexing conventions are used for the same values due to the fact that indexing by name of sensor variable does not change with the model definition, while numerical indexing does. For example, for the Kalman filter design with only 2 sensors, producing v_x and v_y, we would have $\mathbf{R}$ be a 2-by-2 matrix with diagonal elements as R_{v_x} and R_{v_y}, which is, however, different from the above numerical indexing, namely R_{11} and R_{22}. The non-diagonal elements of both matrices, $\mathbf{Q}$ and $\mathbf{R}$, are set to be zero: i.e. $Q_{ij} = 0$ and $R_{ij} = 0$ for $i \neq j$. The measurement noise determines the precision with which the related parameters are determined by the system.

In addition to the GPS and orientation sensors, we assume availability of the speedometer as well as accelerometer on the vehicle. Either the speedometer or the accelerometer provides values of the respective parameter along the vehicle orientation vector, thus producing time series for vectors $\vec{v}(t)$ and $\vec{a}(t)$. In our architectural designs we choose the set of state variables to be x, y, v_x, v_y, a_x, a_y, which differs from that of sensor variables x, y, v, a, θ. To compensate for the difference, we merge the data streams of v and θ via conversion circuits as it is shown in Figure 3.4,a to convert streams of v and θ into that of v_x and v_y for further processing. This is done to reduce noise, remove and correct errors, and fill in missing or lost data. Such conversion is equivalent to mathematical operations $v_x = v \cdot \cos\theta$ and $v_y = v \cdot \sin\theta$. After that, the processed streams of state variables can be stored, while initial data streams of v and θ can be restored, if necessary, with the help of the circuits shown in Figure 3.4,b, which implements mathematical operations $v = \sqrt{v_x^2 + v_y^2}$ and $\theta = \arctan(v_y/v_x)$. The same approach is applied to the stream of

acceleration values a, which are merged with θ into that of $a_x = a \cdot \cos\theta$ and $a_y = a \cdot \sin\theta$ for further processing and the possibility of restoration afterwards to $a = \sqrt{a_x^2 + a_y^2}$ and $\theta = \arctan(a_y/a_x)$. Thus, in our design we process data streams, which are streams of state variables and are x, y, v_x, v_y, a_x, a_y.

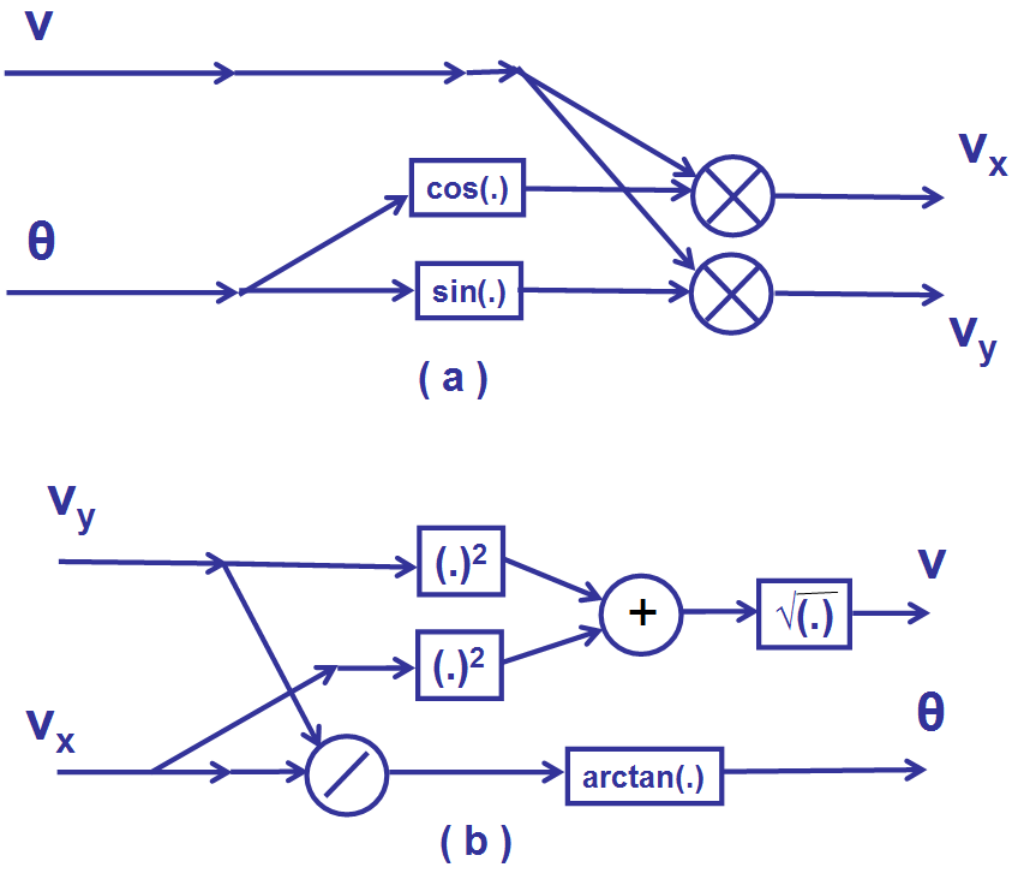


Figure 3.4: Data streams conversion: (a) v and θ streams conversion into v_x and v_y streams; and (b) reversal of v_x and v_y streams back into v and θ ones.

The two-dimensional equation of motion we rely upon in our designs can therefore be written as

$$\begin{cases} x(t+1) = x(t) + v_x(t) \cdot \Delta t + a_x(t) \cdot (\Delta t)^2/2 \\ y(t+1) = y(t) + v_y(t) \cdot \Delta t + a_y(t) \cdot (\Delta t)^2/2 \end{cases} \tag{3.1}$$

where t is time and Δt is a discrete time step.

The state vector $\mathbf{S}$ is

$$\mathbf{S} = \begin{bmatrix} x & y & v_x & v_y & a_x & a_y \end{bmatrix}^T \tag{3.2}$$

and the state transition matrix used is

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0.1 & 0 & 0.01/2 & 0 \\ 0 & 1 & 0 & 0.1 & 0 & 0.01/2 \\ 0 & 0 & 1 & 0 & 0.1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0.1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

When all sensors are operating, i.e. when we observe all 6 state variables x, y, v_x, v_y, a_x and a_y, the sensors observation matrix $\mathbf{C}$ takes the following full form $\mathbf{C}^{(\mathbf{F})}$:

$$\mathbf{C}^{(\mathbf{F})} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

which is actually a form of identity matrix and here it is reflective of the fact that we have chosen all state variables to be observable.

When GPS is the only source of sensory information, so that we observe only latitude x and longitude y, the sensors observation matrix $\mathbf{C}$ becomes the reduced form $\mathbf{C}^{(\mathbf{G})}$:

$$\mathbf{C}^{(\mathbf{G})} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.5}$$

and we call the respective KF operation "GPS-mode" marking it with upper index "(G)" for the observation matrix. Similarly, when the GPS signal is out and we can only rely on v_x and v_y sensory streams, the observation matrix $\mathbf{C}$ reads as the following $\mathbf{C}^{(\mathbf{V})}$ form:

$$\mathbf{C}^{(\mathbf{V})} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.6}$$

and we call the respective KF operation "V-mode", indicating that only velocity components v_x and v_y are observable (the upper index "(V)" for the observation matrix denotes this). Similarly, when 4 streams are available, including that of the GPS for x and y, as well as V-streams, which are v_x and v_y, then we call it "GPS/V-mode" with observation matrix $\mathbf{C}$ as $\mathbf{C}^{(\mathbf{GV})} = \begin{bmatrix} \mathbf{C}^{(\mathbf{G})} & \mathbf{C}^{(\mathbf{V})} \end{bmatrix}^T$ or explicitly:

$$\mathbf{C}^{(\mathbf{GV})} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.7}$$

The architectures presented herewith are targeting cases and operations outlined in Table 3.3 on page 26 and by design are relying mostly on the interaction between two

versions of KF. The first one is operating in GPS/V-mode and thus employing the $\mathbf{C}^{(\mathbf{GV})}$ as its observation matrix defined by equation 3.7, and the second one is in V-mode with $\mathbf{C}^{(\mathbf{V})}$ observation matrix as defined by equation 3.6.

We call the first KF the GPS/V-KF and Algorithm 1 on page 34 presents the logistic of its operation, which is a standard algorithm. Algorithm 1 requires vector $\mathbf{Z}^{(\mathbf{GV})} = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^T$ as an input, where components contain Gaussian noise determined by matrices Q at process level and R at measurement level. The time constant $1/\Delta t^{(GV)}$ defines the rate of the input and it has been varied in experiments to assume the values of 0.01, 0.1 or 1 second to provide the rates of 10, 1 and 0.1 Hz respectively. In our simulations, we always assigned the GPS signal the lowest input rate, which was lower or equal to that of the V-components. At the initialization stage the starting point has been defined for $\mathbf{P}^{(\mathbf{GV})}$ matrix as the unity matrix $\mathbb{1}$ (all-ones), and the initial state is initialized to satisfy equation 2.16. The main algorithmic loop contains two stages of the Kalman filter algorithm: the Update stage, followed by the Predict stage, as described in section 2.3.1. Entry into the Update stage is conditioned by the arrival of the measurement vector $\mathbf{Z}^{(\mathbf{GV})}$ at a set time, which means that if the vector $\mathbf{Z}^{(\mathbf{GV})}$ fails to be delivered in full at a due time mark, then the Update stage is skipped and the predicted state vector $\widehat{\mathbf{S}}^{(\mathbf{GV})}$ stays for the next iteration without correction. Additionally, the error covariance matrix $\mathbf{P}^{(GV)}$ would skip its correction and typically it would result in error increase for the evaluated by GPS/V-KF elements of state vector $\widehat{\mathbf{S}}^{(\mathbf{GV})}$, which in accordance to equation 3.2 is

$$\widehat{\mathbf{S}}^{(\mathbf{GV})} = \begin{bmatrix} \hat{x}^{(GV)} & \hat{y}^{(GV)} & \hat{v}_x^{(GV)} & \hat{v}_y^{(GV)} & \hat{a}_x^{(GV)} & \hat{a}_y^{(GV)} \end{bmatrix}^T \tag{3.8}$$

where the "hat" over the variable means "evaluated version" and the upper index "(GV)" indicates that evaluation is done by GPS/V-KF.

Similarly, the second KF, which we call the V-KF and describe its functionality with Algorithm 2 on page 35, is also a standard algorithm. It is distinguished from the Algorithm 1 by the variables used, rather than algorithmic steps, which also repeat the standard set of equations 2.11 through 2.15 for the Kalman filter algorithm. Here the input vector $\mathbf{Z}^{(\mathbf{V})} = \begin{bmatrix} v_x & v_y \end{bmatrix}^T$ is delivered at the rate $1/\Delta t^{(V)}$, which for this study has always been kept at 10 Hz. The V-KF evaluates its own version of state vector $\widehat{\mathbf{S}}^{(\mathbf{V})}$, which in line with equation 3.2 is:

$$\widehat{\mathbf{S}}^{(\mathbf{V})} = \begin{bmatrix} \hat{x}^{(V)} & \hat{y}^{(V)} & \hat{v}_x^{(V)} & \hat{v}_y^{(V)} & \hat{a}_x^{(V)} & \hat{a}_y^{(V)} \end{bmatrix}^T \tag{3.9}$$

where upper index "(V)" links the variable to the V-KF.

Individually, the two versions of the above KFs, the GPS/V KF as well as V-KF, are quite capable of reducing the Gaussian noise of their data streams with a built-in functionality of the KF algorithm suitable to address task (i) of Table 3.3. However, creating algorithmic interaction between the two allows realization of additional functionalities, such as events detections (ii) through (iv) mentioned in Table 3.3 on page 26. We attribute such new functionalities to the created composition of the two KFs, which supports our declared concept of the compositional KF. In the following sections, these functionalities, which include detection of external events such as error detection, in particular GPS outage as well as GPS offset, are studied in more detail.

Algorithm 1 GPS/V-KF standard algorithm implementation

Require: data in are $\mathbf{Z^{(GV)}} = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^T$ at a synchronized rate of $1/\Delta t^{(GV)}$

Ensure: Estimation of a state vector $\mathbf{S^{(GV)}} = \begin{bmatrix} x & y & v_x & v_y & a_x & a_y \end{bmatrix}^T$ at the k-th time step $\widehat{\mathbf{S}}^{(\mathbf{GV})}(k)$

% Initialization and initial conditions

$k \leftarrow 0$

$\mathbf{P^{(GV)}} \leftarrow \mathbb{1}$

$\widehat{\mathbf{S}}^{(\mathbf{GV})}(k) \leftarrow [\mathbf{C^{(GV)}}]^{-1} \cdot \mathbf{Z^{(GV)}}(k)$

% Main algorithmic loop

while true **do**

% GPS/V-KF Predict stage

$\widehat{\mathbf{S}}^{(\mathbf{GV})}(k+1) \leftarrow \mathbf{A} \cdot \widehat{\mathbf{S}}^{(\mathbf{GV})}(k)$

$\mathbf{P^{(GV)}} \leftarrow \mathbf{A} \cdot \mathbf{P^{(GV)}} \cdot \mathbf{A}^T + \mathbf{Q^{(GV)}}$

% GPS/V-KF Update stage

if $\mathbf{Z^{(GV)}}(k+1)$ comes in **then**

$\mathbf{G^{(GV)}} \leftarrow \mathbf{P^{(GV)}} \cdot [\mathbf{C^{(GV)}}]^T / \{\mathbf{C^{(GV)}} \cdot \mathbf{P^{(GV)}} \cdot [\mathbf{C^{(GV)}}]^T + \mathbf{R^{(GV)}}\}$

$\widehat{\mathbf{S}}^{(\mathbf{GV})}(k+1) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{GV})}(k+1) + \mathbf{G^{(GV)}} \cdot [\mathbf{Z^{(GV)}}(k+1) - \mathbf{C^{(GV)}} \cdot \widehat{\mathbf{S}}^{(\mathbf{GV})}(k+1)]$

$\mathbf{P^{(GV)}} \leftarrow [\mathbf{I} - \mathbf{G^{(GV)}} \cdot \mathbf{C^{(GV)}}] \cdot \mathbf{P^{(GV)}}$

end if

$k \leftarrow k+1$

end while

Algorithm 2 V-KF standard algorithm implementation

Require: data in are $\mathbf{Z^{(V)}} = \begin{bmatrix} v_x & v_y \end{bmatrix}^T$ at a synchronized rate of $1/\Delta t^{(V)}$

Ensure: Estimation of a state vector $\mathbf{S^{(V)}} = \begin{bmatrix} x & y & v_x & v_y & a_x & a_y \end{bmatrix}^T$ at the k-th time step $\widehat{\mathbf{S}}^{(\mathbf{V})}(k)$

% Initialization and initial conditions

$k \leftarrow 0$

$\mathbf{P^{(V)}} \leftarrow \mathbb{1}$

$\widehat{\mathbf{S}}^{(\mathbf{V})}(k) \leftarrow [\mathbf{C^{(V)}}]^{-1} \cdot \mathbf{Z^{(V)}}(k)$

% Main algorithmic loop

while true **do**

% V-KF Predict stage

$\widehat{\mathbf{S}}^{(\mathbf{V})}(k+1) \leftarrow \mathbf{A} \cdot \widehat{\mathbf{S}}^{(\mathbf{V})}(k)$

$\mathbf{P^{(V)}} \leftarrow \mathbf{A} \cdot \mathbf{P^{(V)}} \cdot \mathbf{A}^T + \mathbf{Q^{(V)}}$

% V-KF Update stage

if $\mathbf{Z^{(V)}}(k+1)$ comes in **then**

$\mathbf{G^{(V)}} \leftarrow \mathbf{P^{(V)}} \cdot [\mathbf{C^{(V)}}]^T / \{\mathbf{C^{(V)}} \cdot \mathbf{P^{(V)}} \cdot [\mathbf{C^{(V)}}]^T + \mathbf{R^{(V)}}\}$

$\widehat{\mathbf{S}}^{(\mathbf{V})}(k+1) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{V})}(k+1) + \mathbf{G^{(V)}} \cdot [\mathbf{Z^{(V)}}(k+1) - \mathbf{C^{(V)}} \cdot \widehat{\mathbf{S}}^{(\mathbf{V})}(k+1)]$

$\mathbf{P^{(V)}} \leftarrow [\mathbf{I} - \mathbf{G^{(V)}} \cdot \mathbf{C^{(V)}}] \cdot \mathbf{P^{(V)}}$

end if

$k \leftarrow k+1$

end while

3.4 Architectures for Imputation of Intermittent Data

The need for intermittent data imputation occurs in the IoT system mostly due to the differences in frequencies of various sensors, which are subjected to joint analysis or fusion due to intrinsic connections or relations in the system. For navigational data, the customary GPS signal rates are 1 Hz and 10 Hz. The rates for speedometers may be much higher, such as 1 kHz in digital mode. For analog type circuits, the digitizing of the output can be applied at practically any desired rate. Thus, even in the continuous presence of the GPS signal, the GPS estimates of the location coordinates need to be supplemented by some number of intermediate points in order to convert the initial low rate stream into higher rate stream when such queries occur. For such conversion we explore the following architectures:
1) the regular GPS/V-KF;
2) the regular V-KF;
3) the compositional architecture interactively interfacing the above two cases.

The simulation results conducted for all three designs in Chapter 4 will confirm that there are realistic circumstances in which compositional architecture may offer advantages over individual KF approaches. In brief, such circumstances occur when the GPS signal is unstable, changing measurement noise level and at times causing performance deterioration. In this case, the compositional architecture may limit the impact of the GPS's unstable noise by limiting usage of GPS information to only those windows with low GPS noise, if any, to extract correcting integration constants for its V-KF part. Reliance in generating the required data streams of location coordinates in this case will be mostly on V-KF, but the negative property of the V-KF in accumulating integration phase shift may be compensated by even rare appearance of low noise GPS windows.

3.4.1 Traditional Approach to Data Imputation with KFs

The Kalman filter algorithm has an intrinsic tolerance mechanism in coping with missing measurement data, which has naturally been adopted by scientific community as a lost data replacement procedure on-the-fly. Specifically, as the algorithm's two consecutive stages — Prediction stage and Update stage — are not connected via any dependence, but rather relate via determination of the Kalman Gain coefficient, if the measurement is missing, then it is simply replaced by the prediction value. So, rather than updating with the value, which is a most trustful compromise between measurement and prediction, in the case of missing measurement, the prediction goes unchallenged and takes the place of the update value, thus filling in missing data. The same scenario is applied for lost data. When the measurement is lost, the update stage is reinstating the predicted value as the update value. Similarly, intermittent data are filled in with predicted values as reinstated to become update values in the absence of incoming measurements. The algorithm in this scenario is only able to evaluate the occurred error only at a time when the actual measurement comes in, in which case the error serves as a measure of prediction correctness to eventually adjust the Kalman Gain coefficient and thus train the filter.

The obvious disadvantage of this approach is that it remains blind to actual changes, which system encounters, such as changing conditions of operation, until the measurement occurs. Until then, it acts as a linear extrapolation of the parameter (for linear systems) and carries all related pros and cons, the pros being simplicity and speed, and the cons being reduced reliability and accuracy.

3.4.2 Traditional V-Mode for Data Imputation

We define V-mode of the KF as its operation when the velocity vector data $\vec{v}(t)$ are provided as input, either in the form of its components v_x and v_y, or alternatively as amplitude $v = \sqrt{v_x^2 + v_y^2}$ and phase $\theta = \arctan(v_y/v_x)$ of the $\vec{v}(t)$ vector, which are mutually convertible options as shown in Figure 3.4,a,b.

Specifically, for x and y coordinates, the streams $v-x, v_y$ offer redundancy of derivation by integration:

$$x = \int_{t_0}^{t} v_x(t)dt + C_{x0}, \tag{3.10}$$

$$y = \int_{t_0}^{t} v_y(t)dt + C_{y0}, \tag{3.11}$$

where C_{x0} and C_{y0} are constants defined by the initial condition, that for $t = t_0$ the equations 3.10 and 3.11 bring about the starting point, i.e. $x(t_0) = x_0$ and $y(t_0) = x_0$, from which $C_{x0} = x_0$ and $C_{y0} = y_0$ and thus 3.10 with 3.11 yields

$$x = x_0 + \int_{t_0}^{t} v_x(t)dt, \tag{3.12}$$

$$y = y_0 + \int_{t_0}^{t} v_y(t)dt. \tag{3.13}$$

Supplementing 3.12 and 3.13 with derivation of a_x and a_y components by differentiation of v_x and v_y, i.e.

$$a_x = \frac{dv_x}{dt} \tag{3.14}$$

$$a_y = \frac{dv_y}{dt} \tag{3.15}$$

we gather that v_x and v_y streams are sufficient to derive all missing components of the motion equation to restore the trajectory. Therefore V-KF, i.e. the Kalman filter with vector of velocity as input, should be capable of restoring all other relevant navigational data streams, sufficient to follow the trajectory associated with the motion of the vehicle.

The KF operating in V-mode (the V-KF) in our design has sensors matrix $\mathbf{C}^{(\mathbf{V})}$ defined by equation 3.6 with 1s only in two positions, C_{13} and C_{24}, meant to actualize v_x and v_y.

The rest of the elements are zeros indicating the absence of a GPS signal for x and y (C_{11} and C_{22}) and absence of accelerometer components a_x and a_y data (C_{15} and C_{26}).

Therefore, the error in V-mode will only be perceived for the v-components of the state vector as other components will zero out after multiplication with matrix C. Therefore, the training of the V-KF will aim to minimize the error for predicting the v-components, no matter how it affects coordinates x, y. So, all the errors in predicting the v-components v_x and v_y at the initial stage will be accumulated at the integration stage for coordinates as an overall shift in location, which we refer to as the phase shift for the vehicle's trajectory. After, and if the KF converges, the phase shift would stop accumulating, indicating that optimal gain has been reached, which provides average error in v_x and v_y predictions close to zero. This no longer contributes to accumulation of the phase shift in the coordinates domain via integration. In this way, the phase shift accumulated thus far gets saturated and the KF would preserve the shape of the trajectory.

The addition of the acquired phase shift values $\Delta x_0, \Delta y_0$ to the actual values of the coordinates x_{actual}, y_{actual} can be described as:

$$\begin{cases} x_V = x_{actual} + \Delta x_0 \\ y_V = y_{actual} + \Delta y_0 \end{cases} \tag{3.16}$$

with x_V, y_V being the values generated by V-KF at the update stage of the algorithm and thus are perceived by V-KF trajectory coordinates yielded as part of the generated output streams to the system. Due to its nature, the V-mode is indifferent to the existence of the phase shift in the coordinate streams and therefore the detection of the acquired phase shift is not possible within the V-mode only, but requires additional information from external sensors (for example, GPS external sensors). However, the above consideration of the phase shift acquired during the training cycles of V-mode and otherwise keeping the accuracy for the after-training time frame implies the possibility of extending the improved accuracy into coordinate domain by providing the additional sensory information to the KF to compensate the shift acquired in the training phase in V-mode. This proposition sets the ground for the concept of GPS-pins for V-KF, when such additional information is obtained from the GPS related coordinate streams, which is considered in more detail in the following subsection 3.4.3.

3.4.3 Architecture with GPS Pins for V-Mode

To extend the accuracy provided to v-components, v_x and v_y, by V-KF to coordinate components, x and y, we propose that the absolute value of the coordinates is periodically verified by the GPS signal, whose values $x^{(GV)}, y^{(GV)}$ are assumed to have no phase shift for coordinate components. In this case, the GPS values for coordinates $x^{(GV)}, y^{(GV)}$, which are free from the phase shift, are compared to those $x^{(V)}, y^{(V)}$ supplied by V-KF obtained in V-mode and presumably having the acquired phase shift $\Delta x_0, \Delta y_0$ parts contributing to the values of the coordinates:

$$\begin{cases} x^{(V)} = x^{(GV)} + \Delta x_0 \\ y^{(V)} = y^{(GV)} + \Delta y_0 \end{cases} \tag{3.17}$$

The obtained equations thus allow us to estimate the phase shift values $\Delta x_0, \Delta y_0$:

$$\begin{cases} \Delta x_0 = x^{(V)} - x^{(GV)} \\ \Delta y_0 = y^{(V)} - y^{(GV)} \end{cases} \tag{3.18}$$

and exclude them from the coordinate values of V-KF, placing the trajectory to the GPS-defined position for the specific time moment rather than that calculated in V-mode by the integration step. The concept of the algorithm here is to correct, when the GPS signal allows it, the coordinate values in $\widehat{\mathbf{S}}$, the estimate made by the KF for the state vector of the vehicle:

$$\widehat{\mathbf{S}} = \begin{bmatrix} \hat{x} & \hat{y} & \hat{v}_x & \hat{v}_y & \hat{a}_x & \hat{a}_y \end{bmatrix}^T \tag{3.19}$$

where the hat-symbol over the variable name indicates the estimated by KF value of that variable (i.e. $\widehat{\mathbf{S}}$ is estimated value of $\mathbf{S}$, $\hat{x}$ is estimated value of x and so on for the rest of variables). The relevant architectural design of the interaction between V-KF and GPS/V-KF is shown in Figure 3.5, where it is seen that input streams are x, y, v_x and v_y variables, observation and processing of which result in the output of the corrected version of the state vector estimate $\widehat{\mathbf{S}}^*$. This, in turn, is the result of merging two estimates — one from the GPS/V-KF and another one from the V-KF.

It is important to note here that there are two modules of the KF, which are essentially two separate KFs. In the hardware implementation of the compositional KF, this may be represented by two separate units of two KFs running simultaneously with separate intake. The first KF takes in all four incoming streams, but only at time moments when all four incoming values are available, i.e. GPS-based x and y accompanied by the velocity component streams v_x and v_y, hence our reason for calling it GPS/V-KF. The second KF takes in only velocity components v_x and v_y from the available four streams, regardless of whether the GPS signal is available or not and we call it V-KF. The "if/else"-splitter regulates supply of the streams to the GPS/V-KF, assuring that the streams are passing through only if all four values of variables are available. The V-KF gets the feeding in of its two intake streams of v_x and v_y continuously (i.e. each time the intake is scheduled to take place, for each iteration of the V-KF). The GPS/V-KF is relying on $\mathbf{C}^{(\mathbf{GV})}$ version of observation matrix given by equation 3.7, while V-KF operates via $\mathbf{C}^{(\mathbf{GV})}$ given by equation 3.6. The state matrix A used is the same for both KFs and is given by equation 3.3, which means that both KFs are relying on the same state transition laws. Each of these two KF is running in parallel, i.e. occupying dedicated threads, and therefore by architectural design each KF has its own set of matrices, such as error covariance matrix $\mathbf{P}$ and Kalman gain $\mathbf{G}$ matrix. The GPS/V-KF is generating its estimate of the state vector $\widehat{\mathbf{S}}$, namely $\widehat{\mathbf{S}}^{(\mathbf{GV})}$ given by equation 3.8, while V-KF is producing its version, which is the $\widehat{\mathbf{S}}^{(\mathbf{V})}$ given by equation 3.9.

The resulting estimate of the state vector $\widehat{\mathbf{S}}$ is obtained by merging $\widehat{\mathbf{S}}^{(\mathbf{GV})}$ and $\widehat{\mathbf{S}}^{(\mathbf{V})}$ in a sense of replacing in the $\widehat{\mathbf{S}}^{(\mathbf{V})}$ values of $\hat{x}^{(V)}$ and $\hat{y}^{(V)}$ with respective values of $\hat{x}^{(GV)}$ and $\hat{y}^{(GV)}$ from $\widehat{\mathbf{S}}^{(\mathbf{GV})}$ to construct $\widehat{\mathbf{S}}^*$ as following:

$$\widehat{\mathbf{S}}^* = \begin{bmatrix} \hat{x}^{(GV)} & \hat{y}^{(GV)} & \hat{v}_x^{(V)} & \hat{v}_y^{(V)} & \hat{a}_x^{(V)} & \hat{a}_y^{(V)} \end{bmatrix}^T \tag{3.20}$$

It is at the merging unit in Figure 3.5 where corrections into the state vector estimate are introduced. The essential point here is that the corrections take place only for the cycles with GPS signals coming in, otherwise the V-KF version $\widehat{\mathbf{S}}^{(\mathbf{V})}$ stands as is and proceeds as the output. In either case, the output signal will be taken as the replacement of the update state vector channel for V-KF, which is blocked in a regular sense of stand alone V-KF (as the crossing out arrow on Figure 3.5 indicates).

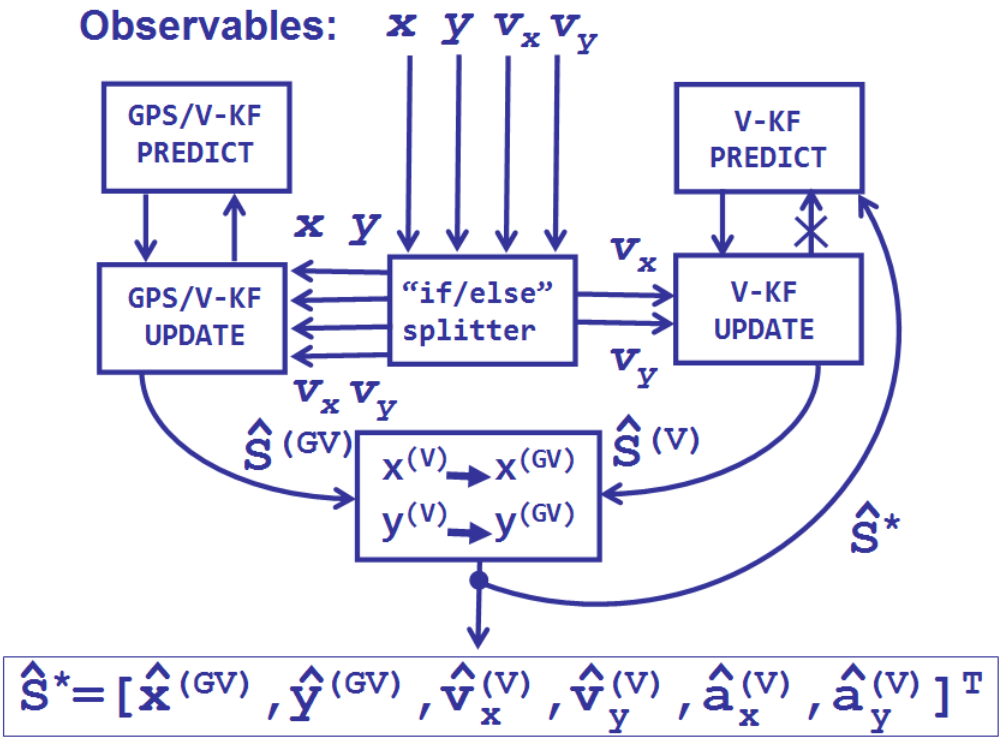


Figure 3.5: Architecture with interaction between GPS/V-KF and V-KF to assist KF in V-mode with GPS pins to compensate for the phase shift acquired in the V-KF training cycle. The operational logistic of the software implementation of this architecture is presented as Algorithm 3 on page 42.

When the value of $\widehat{\mathbf{S}}^{(\mathbf{GV})}$ is generated in the GPS/V-KF unit, it is the result of the regular KF-algorithmic update cycle with the four measured values of the variables, namely x, y, v_x and v_y. This means that the coordinate values $\hat{x}^{(V)}$ and $\hat{y}^{(V)}$ predicted by V-KF for the time moments of comparisons with GPS-originated values are simply discarded and replaced by the values of $\hat{x}^{(GV)}$ and $\hat{y}^{(GV)}$ estimated by GPS/V-KF. In this procedure there is no fusion of the data, but rather priority for a single moment of GPS signal arrival is delegated to the GPS/V-KF to compensate for the phase shift, which is otherwise assumed to be present due to the training cycle of the V-mode operation of the KF. We call this approach "pinning coordinate with GPS" as the V-mode of the KF operation remains intact, supplemented with the detection of the training cycle of V-mode and corrective action of trajectory phase shift compensation.

For the above consideration, Algorithm 3 on page 42 presents the operational logistic of the software implementation of the architecture shown in Figure 3.5 on page 40. Operation of this algorithm requires four data streams coming in, defined by the variables x, y, v_x and v_y. From these four streams the input vectors for the KFs involved are formed, namely $\mathbf{Z^{(V)}} = \begin{bmatrix} v_x & v_y \end{bmatrix}^T$ for V-KF and $\mathbf{Z^{(GV)}} = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^T$ for GPS/V-KF. The rates of $\mathbf{Z^{(V)}}$ and $\mathbf{Z^{(GV)}}$ differ, being $1/\Delta t^{(V)}$ versus $1/\Delta t^{(G)}$ with the input rate for V-KF standing as multiple of the input rate for GPS/V-KF:

$$1/\Delta t^{(V)} = M \cdot 1/\Delta t^{(G)} \tag{3.21}$$

with $M = 1, 2, 3, ...$, where $M = 1$ represents the case of equal rate for all inputs, or in practical terms, when all data are coming in at the same rate (in the examples of this study the rate of 10 Hz for all data components has been considered). If M increases, the relative rate of the GPS signal is reducing, so that, for example, by keeping the rate of incoming measurements for v-components v_x and v_y as 10 Hz, the value $M = 10$ would ensure modelling of GPS streaming of x and y components at 1 Hz. Other conditions being equal, the modelling rate of 0.1 Hz for the GPS signal requires the value of $M = 100$. The above values are used in Chapter 4 in verification experiments.

Algorithm 3 on page 42 aims to eliminate integration error from the estimates $\widehat{\mathbf{S}}^{(\mathbf{V})}$ made by V-KF. For that purpose, the variable *pinFlag*, which initially is set to Boolean value $false$, acquires *true* value inside the conditional implementation of the GPS/V-KF update cycle. Under *true* value of *pinFlag* the "pinning"procedure takes place, in which V-KF overrides its own estimates of updated coordinates $\hat{x}^{(V)}$ and $\hat{y}^{(V)}$ with the ones supplied by GPS/V-KF, $\hat{x}^{(GV)}$ and $\hat{y}^{(GV)}$. As a result, the V-KF accepts in for its update cycle the modified version of the state vector with coordinates from GPS/V-KF version of the update for the state vector. This modified version of the state vector is also used as the overall output of the compositional architecture of joint KF, as depicted in Figure 3.5 (page 40).

Algorithm 3 V-KF adjustment by GPS pin from GPS/V-KF

Require: input data are $\mathbf{Z}^{(\mathbf{V})} = \begin{bmatrix} v_x & v_y \end{bmatrix}^T$ at a rate of $1/\Delta t^{(V)}$ as well as $\mathbf{Z}^{(\mathbf{GV})} = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^T$ at a rate of $1/\Delta t^{(G)}$, with $1/\Delta t^{(V)} = M \cdot 1/\Delta t^{(G)}$, where $M = 1, 2, 3, ...$

Ensure: Excludes accumulated integration error from estimation of a state vector $\mathbf{S}(k) = \begin{bmatrix} x & y & v_x & v_y & a_x & a_y \end{bmatrix}^T$ at the k-th time step $\widehat{\mathbf{S}}(k)$

$k,\ g \leftarrow 0$
$\mathbf{P}^{(\mathbf{GV})},\ \mathbf{P}^{(\mathbf{V})} \leftarrow \mathbb{1}$
$pinFlag \leftarrow false$
$\widehat{\mathbf{S}}^{(\mathbf{GV})}(k) \leftarrow [\mathbf{C}^{(\mathbf{GV})}]^{-1} \cdot \mathbf{Z}^{(\mathbf{GV})}(k)$
$\widehat{\mathbf{S}}^{(\mathbf{V})}(k) \leftarrow [\mathbf{C}^{(\mathbf{V})}]^{-1} \cdot \mathbf{Z}^{(\mathbf{V})}(k)$
while true **do**
% V-KF Predict stage
$\widehat{\mathbf{S}}^{(\mathbf{V})}(k+1) \leftarrow \mathbf{A} \cdot \widehat{\mathbf{S}}^{(\mathbf{V})}(k)$
$\mathbf{P}^{(\mathbf{V})} \leftarrow \mathbf{A} \cdot \mathbf{P}^{(\mathbf{V})} \cdot \mathbf{A}^T + \mathbf{Q}^{(\mathbf{V})}$
% GPS/V-KF Predict stage
if k is equal to $M \times g$ **then**
$\widehat{\mathbf{S}}^{(\mathbf{GV})}(g+1) \leftarrow \mathbf{A} \cdot \widehat{\mathbf{S}}^{(\mathbf{GV})}(g)$
$\mathbf{P}^{(\mathbf{GV})} \leftarrow \mathbf{A} \cdot \mathbf{P}^{(\mathbf{GV})} \cdot \mathbf{A}^T + \mathbf{Q}^{(\mathbf{GV})}$
end if
% V-KF Update stage
if $\mathbf{Z}^{(\mathbf{V})}(k+1)$ comes in **then**
$\mathbf{G}^{(\mathbf{V})} \leftarrow \mathbf{P}^{(\mathbf{V})} \cdot [\mathbf{C}^{(\mathbf{V})}]^T / \{\mathbf{C}^{(\mathbf{V})} \cdot \mathbf{P}^{(\mathbf{V})} \cdot [\mathbf{C}^{(\mathbf{V})}]^T + \mathbf{R}^{(\mathbf{V})}\}$
$\widehat{\mathbf{S}}^{(\mathbf{V})}(k+1) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{V})}(k+1) + \mathbf{G}^{(\mathbf{V})} \cdot [\mathbf{Z}^{(\mathbf{V})}(k+1) - \mathbf{C}^{(\mathbf{V})} \cdot \widehat{\mathbf{S}}^{(\mathbf{V})}(k+1)]$
$\mathbf{P}^{(\mathbf{V})} \leftarrow [\mathbf{I} - \mathbf{G}^{(\mathbf{V})} \cdot \mathbf{C}^{(\mathbf{V})}] \cdot \mathbf{P}^{(\mathbf{V})}$
end if
% GPS/V-KF Update stage
if $\mathbf{Z}^{(\mathbf{GV})}(g+1)$ comes in as GPS signal arrived **then**
$\mathbf{G}^{(\mathbf{GV})} \leftarrow \mathbf{P}^{(\mathbf{GV})} \cdot [\mathbf{C}^{(\mathbf{GV})}]^T / \{\mathbf{C}^{(\mathbf{GV})} \cdot \mathbf{P}^{(\mathbf{GV})} \cdot [\mathbf{C}^{(\mathbf{GV})}]^T + \mathbf{R}^{(\mathbf{GV})}\}$
$\widehat{\mathbf{S}}^{(\mathbf{GV})}(g+1) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{GV})}(g+1) + \mathbf{G}^{(\mathbf{GV})} \cdot [\mathbf{Z}^{(\mathbf{GV})}(g+1) - \mathbf{C}^{(\mathbf{GV})} \cdot \widehat{\mathbf{S}}^{(\mathbf{GV})}(g+1)]$
$\mathbf{P}^{(\mathbf{GV})} \leftarrow [\mathbf{I} - \mathbf{G}^{(\mathbf{GV})} \cdot \mathbf{C}^{(\mathbf{GV})}] \cdot \mathbf{P}^{(\mathbf{GV})}$
$pinFlag \leftarrow true$
else $\{\mathbf{Z}^{(\mathbf{GV})}(g+1)$ did not come in because GPS signal is lost$\}$
$g \leftarrow g+1$
end if
% V-KF accepts GPS pin from GPS/V-KF as correction
if $\{(k+1)$ is equal to $M \times (g+1)$ AND $pinFlag$ is equal $true\}$ **then**
$\widehat{\mathbf{S}}^{(\mathbf{V})}(k+1)[0] \leftarrow \widehat{\mathbf{S}}^{(\mathbf{GV})}(g+1)[0]$
$\widehat{\mathbf{S}}^{(\mathbf{V})}(k+1)[1] \leftarrow \mathbf{S}^{(\mathbf{GV})}(g+1)[1]$
$pinFlag \leftarrow false$
$g \leftarrow g+1$
end if
$\widehat{\mathbf{S}}(k+1) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{V})}(k+1)$
$k \leftarrow k+1$
end while

3.5 Architectures for Missing Data Recovery

The missing data recovery task is somewhat similar to the above considered task of filling in intermittent data. The difference, however, is in the practical situations in which it may occur. A basic scenario for the data to become "missing"is when some malfunctioning occurs in the infrastructure. Examples may include lost packets in the internet network, sensor failure, or other infrastructural failures. For the navigational data streams, a case of interest to consider is GPS outage, which may occur as a satellite signal blockade when the vehicle enters a tunnel or the satellite is shadowed by a mountain. For IoT systems it would be of importance to detect any event of missing data occurrence, and algorithmically adapt to the interruption of the data stream.

3.5.1 Architecture for GPS Outage Handling

The challenge for the KF algorithm here is that there is a natural compensatory mechanism already built-in. Skipping the update stage while replacing missed measurement (i.e. GPS signal) with predicted values may work for only short period of time and — unless the GPS signal returns — leads to the divergence of the filter when the GPS signal fails to recover. In our compositional design, however, there is possibility for the immediate start of the training cycle for the always available V-KF module with the prospect of completely transferring control to V-KF when the GPS signal fails to recover and the GPS-based KF diverges significantly enough. The usually acceptable quantitative criteria for divergence are when the values of the operating parameter exceed beyond the margin of 3 standard deviations. Hence, there are two distinct stages in the GPS outage scenario: when there is GPS signal available and providing data streams of x and y of vehicle positioning, and when there is no GPS signal and the KF must rely solely on v_x and v_y components to monitor the state of the system and in particular, trajectory of the motion. Let us first consider the scenario of unexpected GPS outage , such as a situation of circuits breakdown with signal disappearance. More specifically, in this scenario initially the GPS and velocity sensors are operating synchronously, and thus the GPS related x and y data streams and velocity components v_x and v_y are providing data flow with 10 Hz frequency each. However, at some point the GPS signal is cut off with no recovery and only v_x and v_y data continue to flow in to the rest of the trip. Expectedly, to monitor the state of the system we may rely on GPS/V-KF when the GPS is on but would need to switch to V-KF at the GPS cut off point. To handle this scenario, let us modify the algorithmic structure in Figure 3.5 so that instead of merging two versions of the state vectors we would switch between the two operating modes - the GPS/V-mode when GPS is on and its alternative, the V-mode for GPS outage situations. This would bring us to the schematic of Figure 3.6 where the output switch position is controlled by the status of the GPS signal at the overall input. This switch connects to the output of GPS/V-KF, letting its updated version of the state vector, $\widehat{\mathbf{S}}^{(\mathbf{GV})}$, represent the overall estimate of the state vector, $\widehat{\mathbf{S}}$, if GPS signal is on (case 1). Alternatively, if the GPS signal is off (case of GPS outage, which is case 0 there), the output switch reconnects to the output of V-KF making its estimate $\widehat{\mathbf{S}}^{(\mathbf{V})}$ to represent characterization of the state vector. It is worth noting that unlike in the schematic of

Figure 3.5, the scheme of Figure 3.6 keeps all internal architectural connections intact and thus there is no crossed out connections in Figure 3.6 as in Figure 3.5 for update-to-predict channel.

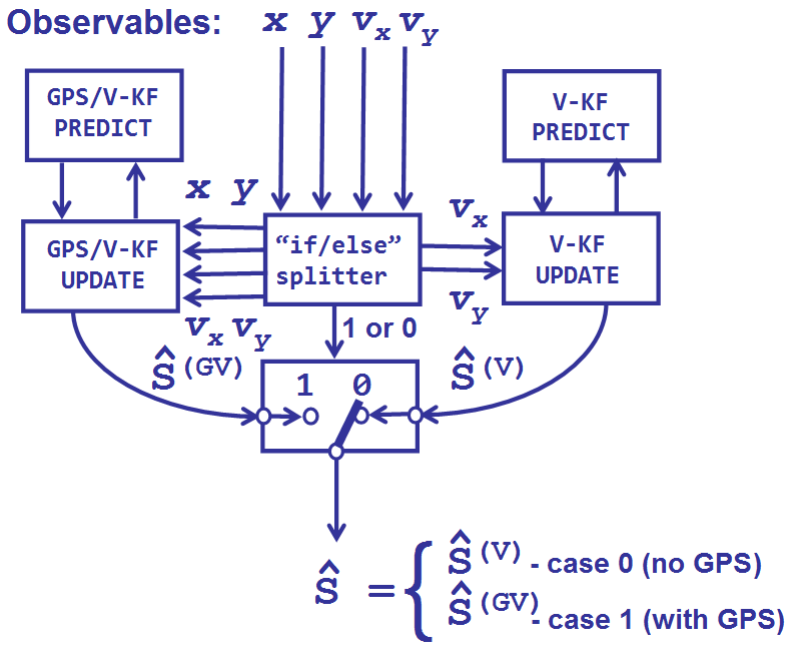


Figure 3.6: Schematic of switching from GPS/V-mode to V-mode KF to handle the GPS outage event. Algorithm 4 on page 46 and Algorithm 5 on page 47 present two viable alternatives of coding logistic of the software implementation for this architecture.

Presented herewith and tested in Chapter 4 are two versions of algorithmic software implementation of the architecture shown in Figure 3.6 on page 44 for handling GPS outage events: (1) serial version, which is Algorithm 4 on page 46, and (2) parallel version, such as Algorithm 5 on page 47. The difference between the two is in the way the GPS/V-KF and the V-KF are operating prior to the occurrence of the GPS outage event. In the serial version, the V-KF is idle prior to GPS outage occurrence, while only GPS/V-KF operates. In the parallel version, the V-KF runs in parallel with the GPS/V-KF at all times, but its output is discarded until the occurrence of the outage event.

More specifically, in Algorithm 4 on page 46 (the serial version) the operation starts with the switch position of "1", when the output generated by the GPS/V-KF is regarded as an overall estimation of the state vector output. The V-KF at this stage keeps idle until the very event of the GPS outage. At that point the switch changes position from "1"to "0"and the operation of the V-KF begins from its initial state, i.e. undergoing training first and then continuing further operation. In the parallel version, i.e. in Algorithm 5 on page 47, the operation also starts with the switch position at "1"and the output generated by the GPS/V-KF counts as an overall estimate. However, the V-KF here does not stay idle, but rather operates in full capacity, passing through a training cycle and continuing thereafter, even though its output is discarded until the detectable occurrence of the GPS

outage. At the point of GPS outage, the switch changes position to "0"and then the output from the V-KF starts counting as an overall estimate, while its counterpart (i.e. the output from the GPS/V-KF) is discarded.

Coding of the serial version differs from that of the parallel version only in one aspect — namely, the conditional protection of the V-KF operation by the value of integer variable $internalBias$, which controls the position of the output switch in Figure 3.6 on page 44 and takes the value "0"or "1"depending on its position mark. Specifically, such conditioning is present in the serial version, allowing entry to execution of V-KF operations only when $internalBias = 0$ is satisfied. In contrast, in the parallel version the execution of V-KF operations is unconditional, and therefore is carried out regardless of the value of $internalBias$. Otherwise, the remainder of the coding for both algorithmic versions is identical, which is seen from the comparison of the coding lines of Algorithm 4 on page 46 and that of Algorithm 5 on page 47.

Both algorithms assume the same measurement rate for all variables, whose delivery is guaranteed, so that the event when the GPS signal fails to arrive would be interpreted as a definite outage. The initial position of the $internalBias$ is "1"meaning that it starts with the switch position at "1"and the GPS signal is present and therefore the output from GPS/V-KF is regarded as an overall output at the beginning. In both code implementation versions, the execution starts by entering the GPS/V-KF set of calculations, restricted by the condition that $internalBias$ is equal "1". As $internalBias$ is initiated at value "1", the condition is guaranteed to be fulfilled for the first run. This portion of the code results in producing $\widehat{\mathbf{S}}^{(\mathbf{GV})}$ vector as a GPS/V-KF's estimate of the state vector. If the GPS signal arrives on time, the follow up code execution is different for serial and parallel versions. For the serial version, code execution proceeds to the stage of final selection of the state estimate vector, while in parallel version this is preceded by mandatory execution of the V-KF cycle.

At the final stage, the selection of the output state vector is made based on the criterion of the $internalBias$ value (i.e. the position of the switch) . At $internalBias = 1$ the $\widehat{\mathbf{S}}^{(\mathbf{GV})}$ goes to overall output as a state estimate vector, while alternatively, i.e. at $internalBias = 0$ the $\widehat{\mathbf{S}}^{(\mathbf{V})}$ does. The time index k increment completes the main loop cycle, which returns execution to the main loop's start for the next cycle.

A GPS outage event is detected by the conditional clause, which is implemented via the "else"link to the GPS/V-KF cycle. This clause is executed in case of GPS signal failure and it irreversibly changes the value of $internalBias$ to "0", which simulates the shift of the switch to position "0". In this case, in the serial version the execution of the V-KF cycle initiates and becomes mandatory from that point onwards, and as such it becomes equivalent to the parallel version where it is always the case. From that moment, the continuation of both versions, serial and parallel one, follow the same route, repeating the main loop cycle through implementation of the V-KF calculations for the state estimate as overall output. This, in turn, signifies the fact that observation control is fully transferred to the V-KF due to the occurrence of a GPS outage event.

Algorithm 4 Serial version for GPS outage event detection and compensation

Require: data $\mathbf{Z^{(V)}} = \begin{bmatrix} v_x & v_y \end{bmatrix}^T$ and $\mathbf{Z^{(GV)}} = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^T$ arrive at the same rate of $1/\Delta t$, but unrecoverable GPS outage at a time step $k = k_{out}$ cancels $\mathbf{Z^{(GV)}}$

Ensure: Recovers convergence in estimation of a state vector $\mathbf{S}(k) = \begin{bmatrix} x & y & v_x & v_y & a_x & a_y \end{bmatrix}^T$ after GPS outage event, i.e. for $k > k_{out}$).

% Initialization and initial conditions

$k \leftarrow 0$

$\mathbf{P^{(GV)}}, \mathbf{P^{(V)}} \leftarrow \mathbb{1}$

$internalBias \leftarrow 1$

$\widehat{\mathbf{S}}^{(\mathbf{GV})}(k) \leftarrow [\mathbf{C}^{(\mathbf{GV})}]^{-1} \cdot \mathbf{Z}^{(\mathbf{GV})}(k)$

while true **do**

  if $internalBias$ is equal 1 **then**

    % GPS/V-KF Predict followed by Update stage

    $\widehat{\mathbf{S}}^{(\mathbf{GV})}(k+1) \leftarrow \mathbf{A} \cdot \widehat{\mathbf{S}}^{(\mathbf{GV})}(k)$

    $\mathbf{P^{(GV)}} \leftarrow \mathbf{A} \cdot \mathbf{P^{(GV)}} \cdot \mathbf{A}^T + \mathbf{Q^{(GV)}}$

    if $\mathbf{Z}^{(\mathbf{GV})}(k+1)$ comes in as GPS signal arrived **then**

      $\mathbf{G^{(GV)}} \leftarrow \mathbf{P^{(GV)}} \cdot [\mathbf{C^{(GV)}}]^T / \{\mathbf{C^{(GV)}} \cdot \mathbf{P^{(GV)}} \cdot [\mathbf{C^{(GV)}}]^T + \mathbf{R^{(GV)}}\}$

      $\widehat{\mathbf{S}}^{(\mathbf{GV})}(k+1) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{GV})}(k+1) + \mathbf{G^{(GV)}} \cdot [\mathbf{Z}^{(\mathbf{GV})}(k+1) - \mathbf{C^{(GV)}} \cdot \widehat{\mathbf{S}}^{(\mathbf{GV})}(k+1)]$

      $\mathbf{P^{(GV)}} \leftarrow [\mathbf{I} - \mathbf{G^{(GV)}} \cdot \mathbf{C^{(GV)}}] \cdot \mathbf{P^{(GV)}}$

    else $\{\mathbf{Z}^{(\mathbf{GV})}(k+1)$ did not come in because GPS signal is lost$\}$

      % V-KF accepts state vector from GPS/V-KF as a pin

      $\widehat{\mathbf{S}}^{(\mathbf{V})}(k) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{GV})}(k)$

      $internalBias \leftarrow 0$

    end if

  end if

  if $internalBias$ is equal 0 **then**

    % V-KF Predict followed by Update stage

    $\widehat{\mathbf{S}}^{(\mathbf{V})}(k+1) \leftarrow \mathbf{A} \cdot \widehat{\mathbf{S}}^{(\mathbf{V})}(k)$

    $\mathbf{P^{(V)}} \leftarrow \mathbf{A} \cdot \mathbf{P^{(V)}} \cdot \mathbf{A}^T + \mathbf{Q^{(V)}}$

    if $\mathbf{Z}^{(\mathbf{V})}(k+1)$ comes in **then**

      $\mathbf{G^{(V)}} \leftarrow \mathbf{P^{(V)}} \cdot [\mathbf{C^{(V)}}]^T / \{\mathbf{C^{(V)}} \cdot \mathbf{P^{(V)}} \cdot [\mathbf{C^{(V)}}]^T + \mathbf{R^{(V)}}\}$

      $\widehat{\mathbf{S}}^{(\mathbf{V})}(k+1) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{V})}(k+1) + \mathbf{G^{(V)}} \cdot [\mathbf{Z}^{(\mathbf{V})}(k+1) - \mathbf{C^{(V)}} \cdot \widehat{\mathbf{S}}^{(\mathbf{V})}(k+1)]$

      $\mathbf{P^{(V)}} \leftarrow [\mathbf{I} - \mathbf{G^{(V)}} \cdot \mathbf{C^{(V)}}] \cdot \mathbf{P^{(V)}}$

    end if

  end if

  % Final selection of the output state vector

  if $internalBias$ is equal 1 **then**

    $\widehat{\mathbf{S}}(k+1) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{GV})}(k+1)$

  else $\{$ $internalBias$ is equal 0 $\}$

    $\widehat{\mathbf{S}}(k+1) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{V})}(k+1)$

  end if

  $k \leftarrow k+1$

end while

Algorithm 5 Parallel version for GPS outage event detection and compensation

Require: data $\mathbf{Z^{(V)}} = \begin{bmatrix} v_x & v_y \end{bmatrix}^T$ and $\mathbf{Z^{(GV)}} = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^T$ arrive at the same rate of $1/\Delta t$, but unrecoverable GPS outage at a time step $k = k_{out}$ cancels $\mathbf{Z^{(GV)}}$

Ensure: Recovers convergence in estimation of a state vector $\mathbf{S}(k) = \begin{bmatrix} x & y & v_x & v_y & a_x & a_y \end{bmatrix}^T$ after GPS outage event, i.e. for $k > k_{out}$).

% Initialization and initial conditions

$k \leftarrow 0$

$\mathbf{P^{(GV)}}, \mathbf{P^{(V)}} \leftarrow \mathbb{1}$

$internalBias \leftarrow 1$

$\widehat{\mathbf{S}}^{(\mathbf{GV})}(k) \leftarrow [\mathbf{C}^{(\mathbf{GV})}]^{-1} \cdot \mathbf{Z}^{(\mathbf{GV})}(k)$

while true **do**

  if $internalBias$ is equal 1 **then**

    % GPS/V-KF Predict followed by Update stage

    $\widehat{\mathbf{S}}^{(\mathbf{GV})}(k+1) \leftarrow \mathbf{A} \cdot \widehat{\mathbf{S}}^{(\mathbf{GV})}(k)$

    $\mathbf{P}^{(\mathbf{GV})} \leftarrow \mathbf{A} \cdot \mathbf{P}^{(\mathbf{GV})} \cdot \mathbf{A}^T + \mathbf{Q}^{(\mathbf{GV})}$

    if $\mathbf{Z}^{(\mathbf{GV})}(k+1)$ comes in as GPS signal arrived **then**

      $\mathbf{G}^{(\mathbf{GV})} \leftarrow \mathbf{P}^{(\mathbf{GV})} \cdot [\mathbf{C}^{(\mathbf{GV})}]^T / \{\mathbf{C}^{(\mathbf{GV})} \cdot \mathbf{P}^{(\mathbf{GV})} \cdot [\mathbf{C}^{(\mathbf{GV})}]^T + \mathbf{R}^{(\mathbf{GV})}\}$

      $\widehat{\mathbf{S}}^{(\mathbf{GV})}(k+1) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{GV})}(k+1) + \mathbf{G}^{(\mathbf{GV})} \cdot [\mathbf{Z}^{(\mathbf{GV})}(k+1) - \mathbf{C}^{(\mathbf{GV})} \cdot \widehat{\mathbf{S}}^{(\mathbf{GV})}(k+1)]$

      $\mathbf{P}^{(\mathbf{GV})} \leftarrow [\mathbf{I} - \mathbf{G}^{(\mathbf{GV})} \cdot \mathbf{C}^{(\mathbf{GV})}] \cdot \mathbf{P}^{(\mathbf{GV})}$

    else $\{\mathbf{Z}^{(\mathbf{GV})}(k+1)$ did not come in because GPS signal is lost$\}$

      % V-KF accepts state vector from GPS/V-KF as a pin

      $\widehat{\mathbf{S}}^{(\mathbf{V})}(k) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{GV})}(k)$

      $internalBias \leftarrow 0$

    end if

  end if

  % V-KF Predict followed by Update stage

  $\widehat{\mathbf{S}}^{(\mathbf{V})}(k+1) \leftarrow \mathbf{A} \cdot \widehat{\mathbf{S}}^{(\mathbf{V})}(k)$

  $\mathbf{P}^{(\mathbf{V})} \leftarrow \mathbf{A} \cdot \mathbf{P}^{(\mathbf{V})} \cdot \mathbf{A}^T + \mathbf{Q}^{(\mathbf{V})}$

  if $\mathbf{Z}^{(\mathbf{V})}(k+1)$ comes in **then**

    $\mathbf{G}^{(\mathbf{V})} \leftarrow \mathbf{P}^{(\mathbf{V})} \cdot [\mathbf{C}^{(\mathbf{V})}]^T / \{\mathbf{C}^{(\mathbf{V})} \cdot \mathbf{P}^{(\mathbf{V})} \cdot [\mathbf{C}^{(\mathbf{V})}]^T + \mathbf{R}^{(\mathbf{V})}\}$

    $\widehat{\mathbf{S}}^{(\mathbf{V})}(k+1) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{V})}(k+1) + \mathbf{G}^{(\mathbf{V})} \cdot [\mathbf{Z}^{(\mathbf{V})}(k+1) - \mathbf{C}^{(\mathbf{V})} \cdot \widehat{\mathbf{S}}^{(\mathbf{V})}(k+1)]$

    $\mathbf{P}^{(\mathbf{V})} \leftarrow [\mathbf{I} - \mathbf{G}^{(\mathbf{V})} \cdot \mathbf{C}^{(\mathbf{V})}] \cdot \mathbf{P}^{(\mathbf{V})}$

  end if

  % Final selection of the output state vector

  if $internalBias$ is equal 1 **then**

    $\widehat{\mathbf{S}}(k+1) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{GV})}(k+1)$

  else $\{$ $internalBias$ is equal 0 $\}$

    $\widehat{\mathbf{S}}(k+1) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{V})}(k+1)$

  end if

  $k \leftarrow k+1$

end while

3.6 Architectures for Error Elimination

Since the KF relies on the state equations to estimate next state based on the previous one, it does have criteria to detect sufficiently large deviations from the regular flow of events. Specific quantitative parameters used in the KF algorithm to evaluate whether deviation of the measured value of the variable is within expectation or exceeding it, is located in the measurement covariance matrix $\mathbf{R}$, where diagonal members are the measurement variances. Deviation of any parameter within a three standard deviation range is regarded as a trusted measurement noise. Exceeding this range is detectable and is to be regarded as an unusual event with underlying reasons, such as possible changes to outside conditions, and noise level, which require adjustment of the KF parameters. If the consistent and repeatable change of the noise level is detected, then the $\mathbf{R}$ matrix needs to be updated according to new operating conditions, which is essential for the convergence of the KF algorithm. However, if the deviation of the variable value exceeds the expectation range only once within a large number of sequential measurements, then such an event can qualify as an outlier and simply be discarded. These are trivial occasions for the KF algorithm to deal with. However, the event which at first is viewed as an outlier but continues to persist consistently thereafter ceases to be an outlier and requires more sophisticated processing to be adequately addressed and resolved. An example of such an event is the GPS offset occurrence, the consideration of which we will now proceed to.

3.6.1 Architecture for GPS Offset Compensation

Above the GPS outage has been considered as the event which affects the operation of the KF and compensatory mechanisms have been explored to prevent data loss or alteration. From the same perspective, the occurrence of GPS offset is considered herewith. The GPS offset differs from the outage in that the signal does not disappear but rather acquires a sudden phase shift in the vehicle perceived location $\mathbf{\Delta r}^*$ and coordinate values as its components Δx^* and Δy^*, which exceeds the margin of 3 standard deviations to a somewhat significant extent. This extent depends on the physical cause of the GPS offset, of which the largest could be control segment mistakes due to computer or human error that can cause errors from one meter to hundreds of kilo-metres. Additionally, the receiver errors from software or hardware failures can cause blunder errors of any size.

To account for the impact of the GPS offset $\mathbf{\Delta r}^*$ we write the values containing the offset components Δx^* and Δy^* as

$$\begin{cases} x^* = x + \Delta x^* \\ y^* = y + \Delta y^* \end{cases} \tag{3.22}$$

where x and y are GPS-derived coordinates without offset while x^* and y^* are the same but including the offset contributions. To eliminate the offset contribution, it needs to be deducted from the incoming values. The issue here is how to make an estimate of the offset and distinguish it from the outlier. The suggested solution here is based, on one hand, on the fact that GPS offset does not affect V-KF at all, as the V-KF relies on the input

of V-components, v_x and v_y, which are measured independently from the GPS. Therefore, the offset contributions into GPS coordinate components x and y are not affecting v_x and v_y, which in turn produce estimates of coordinates $\hat{x}^{(V)}$ and $\hat{y}^{(V)}$. On the other hand, the GPS/V-KF takes the values of the offset as the trusted values from the sensors. Thus, the GPS/V-KF will reflect the contribution of the GPS offset into its estimates of coordinates $\hat{x}^{(GV)}$ and $\hat{y}^{(GV)}$. By comparing the two coordinate sets — one from GPS/V-KF and the other from V-KF, — we can deduce the estimate of the GPS offset components $\delta\hat{x}^*$ and $\delta\hat{y}^*$ as following:

$$\begin{cases} \hat{x}^{(GV)} = \hat{x}^{(V)} + \delta\hat{x}^* \\ \hat{y}^{(GV)} = \hat{y}^{(V)} + \delta\hat{y}^* \end{cases} \tag{3.23}$$

and thus

$$\begin{cases} \delta\hat{x}^* = \hat{x}^{(GV)} - \hat{x}^{(V)} \\ \delta\hat{y}^* = \hat{y}^{(GV)} - \hat{y}^{(V)} \end{cases} \tag{3.24}$$

The obtained estimates $\delta\hat{x}^*$ and $\delta\hat{y}^*$ of the GPS components need to be deducted from the GPS input values in order to exclude the offset influence on the GPS/V-KF operation, as shown in Figure 3.7. It is important to note here that the offset of the GPS does qualify as being detected if and only if at least one of the two values, namely $\delta\hat{x}^*$ or $\delta\hat{y}^*$ exceeds some pre-selected threshold values. We name those threshold values here as Γ_x^{th} for the $\delta\hat{x}^*$ parameter and Γ_y^{th} for the $\delta\hat{y}^*$ parameter. In turn, the threshold values should be set higher than at least three standard deviations of the respective variables, i.e. x for Γ_x^{th} and y for Γ_y^{th} in our case.

The code structure of the architecture from Figure 3.7 on page 54 is shown as Algorithm 6 on page 52 supplemented with Algorithm 7 on page 53 as part of its offset detection test. The algorithm allows operation of two constituent KFs at different rates, which in turn are defined by the rates of incoming data: $1/\Delta t^{(G)}$ for GPS data and $1/\Delta t^{(V)}$ for V-components data. There is one restriction applied here: that $1/\Delta t^{(V)}$ is the exact multiple of $1/\Delta t^{(G)}$ as given by the equation 3.21 on page 41.

Threshold parameters Γ_x^{th} and Γ_y^{th} are used to estimate the offsets along x and y axis respectively. The values of threshold parameters are chosen to exceed three standard deviations of the respective measurements: $\Gamma_x^{th} > 3\sqrt{R_{11}}$ and $\Gamma_y^{th} > 3\sqrt{R_{22}}$, where R_{11} and R_{22} are elements of the measurement matrix $\mathbf{R}$.

The variables $\delta\hat{x}^*$ and $\delta\hat{y}^*$ are to accept the detected values of the offset components and to account for its contribution in the phase shifts Δx^* or Δy^* as accumulated offset of GPS values in line with equation 3.22 on page 48. Initially, all these parameters — Δx^*, Δy^*, $\delta\hat{x}^*$ and $\delta\hat{y}^*$ — are set at zero, and the Boolean variables $offsetFlagX$ and $offsetFlagY$ that serve as indicators of occurrences unaccounted for by the GPS offset contributions are set at $false$. Therefore, there is no offset at the starting point of the simulation. The two KFs, the GPS/V-KF and the V-KF, are run in parallel to produce estimates of the state vector $\widehat{\mathbf{S}}^{(\mathbf{GV})}$ and $\widehat{\mathbf{S}}^{(\mathbf{V})}$. After that the routine GPSoffsetDetection()

is called to verify whether there is any detectable offset contributions $\delta\hat{x}^*$ or $\delta\hat{y}^*$ in the GPS data, which need to be accounted for.

The routine GPSoffsetDetection() is described as Algorithm 7 on page 53. It is called in from within the update cycle of the GPS/V-KF as a last instruction, protected by the condition of completeness of the training for all Kalman filters involved, i.e. GPS/V-KF as well as V-KF in our case. This algorithmic conditioning represents functionality of the architectural switches on the feedback lines in Figure 3.7 on page 54, which are included in order to be able to block out the undesirable influence of the GPS offset detection and correction routines on the training of the Kalman filters. The reason for this is that the training of the Kalman filters requires its exposure to the regular conditions of operation under which the convergence needs to be achieved. The detection of special events, such as GPS offset, conceptually is relying on the output from the Kalman filters, which are already trained, and which provide the convergence for the given operational conditions. Here the operational conditions include all aspects of the system operation, including features of the evolution of the state variables as well as noise levels of the sensors and processes. Therefore, it is assumed that any unusual influences are to be avoided during the training cycle for the Kalman filters. This purpose is served by disconnecting the feedback lines in Figure 3.7 on page 54, which is algorithmically represented by the conditional restriction of the access to the GPSoffsetDetection() routine call. The routine is called only when the training of both filters, the GPS/V-KF and the V-KF, is completed and therefore their convergence for the regular operational conditions is achieved.

During the training of the Kalman filters, the routine GPSoffsetDetection() is not accessible and as a result the output state vector $\widehat{\mathbf{S}}$ assumes either the value of $\widehat{\mathbf{S}}^{(\mathbf{GV})}$ when the GPS signal is coming in, or, alternatively, the value of $\widehat{\mathbf{S}}^{(\mathbf{V})}$ between the GPS signal arrivals. In other words, the output state vector $\widehat{\mathbf{S}}$ discards output of the V-KF whenever there is output from the GPS/V-KF, otherwise it assumes the V-KF value.

The situation changes after the training of the Kalman filters is completed, in that the routine GPSoffsetDetection() starts determining which value is assigned to the output state vector $\widehat{\mathbf{S}}$. Whenever the GPS offset is detected on any of the coordinates, x or y, the output from GPS/V-KF, i.e. $\widehat{\mathbf{S}}^{(\mathbf{GV})}$, is discarded and replaced by the output from the V-KF, i.e. $\widehat{\mathbf{S}}^{(\mathbf{V})}$, to represent the overall output value for the time stamps $M \cdot (g+1)$ of GPS signal arrivals. Alternatively, when there is no detectable GPS offset on either coordinate in arrived GPS data, the GPS/V-KF output $\widehat{\mathbf{S}}^{(\mathbf{GV})}$ takes priority in both representing the overall output $\widehat{\mathbf{S}}$ as well as imposing the GPS pin onto the V-KF, as Algorithm 7 on page 53 asserts.

In order to detect the GPS offset, the GPSoffsetDetection() routine applies thresholds Γ_x^{th} and Γ_y^{th} to the differences of the coordinate estimates from vector $\widehat{\mathbf{S}}^{(\mathbf{GV})}$ and vector $\widehat{\mathbf{S}}^{(\mathbf{V})}$ and concludes on the detection of the offset component in case any of these thresholds are broken. The detected offset component, which exceeded the designated threshold, Γ_x^{th} or Γ_y^{th}, is then deducted from the input phase shift. This deduction ensures correction of the offset by compensating and thus eliminating it from the input stream. This compensation is a one-time operation, because as soon as the offset value contribution is accounted for in the value of the phase shift, i.e. $\delta\hat{x}^*$ in Δx^* as well as $\delta\hat{y}^*$ in Δy^*, it will automatically

be applied to all subsequent incoming data for x and/or y.

Additionally, the routine GPSoffsetDetection() updates the record about GPS offset events $\mathbf{E}^{(\mathbf{G})}_{\mathbf{offset}}$, which holds the relevant characteristics of the detected offset events. This includes the time stamp for the event, the sign and the value of the detected offset.

The GPSoffsetDetection() routine ends with the code lines designed to reset the offset detecting variables to their initial state of readiness for the next event detection. The Boolean variables $offsetFlagX$ and $offsetFlagY$ to $false$, and the offset modification variables $\delta\hat{x}^*$ and $\delta\hat{y}^*$ to zero. After that the algorithmic control of the software is returned back to Algorithm 6 on page 52 to continue to the next time slot of arriving data in the main algorithmic loop.

If thresholds for GPS offset detection are chosen correctly, then the offset detection and compensation architecture shown in Figure 3.7 holds a promise to be operational and its performance needs further verification in simulation and modelling experiments. Particular concern is to be given to the training periods for the Kalman filters involved and their interference with the application of the algorithmic routines (i.e. combined algorithm presented herewith as two connected ones — Algorithm 6 on page 52 with Algorithm 7 on page 53). The two switches in the feedback lines of Figure 3.7 aim to allow training periods to be free from interference with these algorithmic operations. The details of these verifications are presented herewith in Chapter 4.

Algorithm 6 GPS offset occurrence detection and compensation.

Require: data in are $\mathbf{Z^{(V)}} = \begin{bmatrix} v_x & v_y \end{bmatrix}^T$ at a rate of $1/\Delta t^{(V)}$ as well as $\mathbf{Z^{(GV)}} = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^T$ at a rate of $1/\Delta t^{(G)}$, with $1/\Delta t^{(V)} = M \cdot 1/\Delta t^{(G)}$, where $M = 1, 2, 3, ...$ Thresholds $\Gamma_x^{th} > 3\sqrt{R_{11}}$ OR $\Gamma_y^{th} > 3\sqrt{R_{22}}$;

Ensure: Detects GPS offset events, accumulates the record of its occurrences $\mathbf{E_{offset}^{(G)}}$, and compensates the GPS offset values

k, g, Δx^*, Δy^*, $\delta\hat{x}^*$, $\delta\hat{y}^* \leftarrow 0$
$\mathbf{P^{(GV)}}$, $\mathbf{P^{(V)}} \leftarrow \mathbb{1}$
$\mathbf{E_{offset}^{(G)}} \leftarrow \emptyset$
$offsetFlagX$, $offsetFlagY \leftarrow false$
$\widehat{\mathbf{S}}^{(\mathbf{GV})}(k) \leftarrow [\mathbf{C}^{(\mathbf{GV})}]^{-1} \cdot \mathbf{Z}^{(\mathbf{GV})}(k)$
$\widehat{\mathbf{S}}^{(\mathbf{V})}(k) \leftarrow [\mathbf{C}^{(\mathbf{V})}]^{-1} \cdot \mathbf{Z}^{(\mathbf{V})}(k)$
while *true* **do**
  % V-KF Predict followed by Update stage
  $\widehat{\mathbf{S}}^{(\mathbf{V})}(k+1) \leftarrow \mathbf{A} \cdot \widehat{\mathbf{S}}^{(\mathbf{V})}(k)$
  $\mathbf{P^{(V)}} \leftarrow \mathbf{A} \cdot \mathbf{P^{(V)}} \cdot \mathbf{A}^T + \mathbf{Q^{(V)}}$
  if $\mathbf{Z}^{(\mathbf{V})}(k+1)$ comes in **then**
    $\mathbf{G^{(V)}} \leftarrow \mathbf{P^{(V)}} \cdot [\mathbf{C^{(V)}}]^T / \{\mathbf{C^{(V)}} \cdot \mathbf{P^{(V)}} \cdot [\mathbf{C^{(V)}}]^T + \mathbf{R^{(V)}}\}$
    $\widehat{\mathbf{S}}^{(\mathbf{V})}(k+1) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{V})}(k+1) + \mathbf{G^{(V)}} \cdot [\mathbf{Z}^{(\mathbf{V})}(k+1) - \mathbf{C^{(V)}} \cdot \widehat{\mathbf{S}}^{(\mathbf{V})}(k+1)]$
    $\mathbf{P^{(V)}} \leftarrow [\mathbf{I} - \mathbf{G^{(V)}} \cdot \mathbf{C^{(V)}}] \cdot \mathbf{P^{(V)}}$
  end if
  if $\{(k+1)$ is equal to $M \cdot (g+1)\}$ **then**
    % GPS/V-KF Predict stage
    $\widehat{\mathbf{S}}^{(\mathbf{GV})}(g+1) \leftarrow \mathbf{A} \cdot \widehat{\mathbf{S}}^{(\mathbf{GV})}(g)$
    $\mathbf{P^{(GV)}} \leftarrow \mathbf{A} \cdot \mathbf{P^{(GV)}} \cdot \mathbf{A}^T + \mathbf{Q^{(GV)}}$
    % GPS/V-KF Update stage with built-in GPS offset compensation
    if $\mathbf{Z}^{(\mathbf{GV})}(g+1)$ comes in as GPS signal arrives **then**
      $\mathbf{x^{(GV)}}(g+1) \leftarrow \mathbf{x^{(GV)}}(g+1) - \Delta x^*$
      $\mathbf{y^{(GV)}}(g+1) \leftarrow \mathbf{y^{(GV)}}(g+1) - \Delta y^*$
      $\mathbf{G^{(GV)}} \leftarrow \mathbf{P^{(GV)}} \cdot [\mathbf{C^{(GV)}}]^T / \{\mathbf{C^{(GV)}} \cdot \mathbf{P^{(GV)}} \cdot [\mathbf{C^{(GV)}}]^T + \mathbf{R^{(GV)}}\}$
      $\widehat{\mathbf{S}}^{(\mathbf{GV})}(g+1) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{GV})}(g+1) + \mathbf{G^{(GV)}} \cdot [\mathbf{Z}^{(\mathbf{GV})}(g+1) - \mathbf{C^{(GV)}} \cdot \widehat{\mathbf{S}}^{(\mathbf{GV})}(g+1)]$
      $\mathbf{P^{(GV)}} \leftarrow [\mathbf{I} - \mathbf{G^{(GV)}} \cdot \mathbf{C^{(GV)}}] \cdot \mathbf{P^{(GV)}}$
      $\widehat{\mathbf{S}}(M \cdot g + M) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{GV})}(g+1)$
      % built-in GPS offset detection and compensation
      if training of Kalman filters is completed **then**
        GPSoffsetDetection();
      end if
    end if
    $g \leftarrow g + 1$
  else
    $\widehat{\mathbf{S}}(k+1) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{V})}(k+1)$
  end if
  $k \leftarrow k + 1$
end while

Algorithm 7 GPSoffsetDetection() routine: a GPS offset detection test called from Algorithm 6 on page 52

Require: access to the data of Algorithm 6 on page 52 to perform GPS offset detection test there within GPSoffsetDetection() routine call

Ensure: completion of GPSoffsetDetection() routine of the Algorithm 6 on page 52 to verify whether the offset thresholds are broken to set up the offset compensating phase shift values Δx^* and Δy^*, as well as fill in the record about GPS offset event $\mathbf{E}^{(\mathbf{G})}_{\mathbf{offset}}$

% GPS offset event detection test as GPSoffsetDetection() call

GPSoffsetDetection(){

% Applying threshold to detect GPS offset event

if $\|\hat{x}^{(GV)} - \hat{x}^{(V)}\| >= \Gamma_x^{th}$ **then**

$\delta\hat{x}^* \leftarrow (\hat{x}^{(GV)} - \hat{x}^{(V)})$

$\Delta x^* \leftarrow \Delta x^* + \delta\hat{x}^*$

$\hat{x}^{(GV)} \leftarrow \hat{x}^{(V)}$

$offsetFlagX \leftarrow true$

end if

if $\|\hat{y}^{(GV)} - \hat{y}^{(V)}\| >= \Gamma_y^{th}$ **then**

$\delta\hat{y}^* \leftarrow (\hat{y}^{(GV)} - \hat{y}^{(V)})$

$\Delta y^* \leftarrow \Delta y^* + \delta\hat{y}^*$

$\hat{y}^{(GV)} \leftarrow \hat{y}^{(V)}$

$offsetFlagY \leftarrow true$

end if

% Updating record by newly detected GPS offset event

if $offsetFlagX$ is equal $true$ OR $offsetFlagY$ is equal $true$ **then**

$\mathbf{E}^{(\mathbf{G})}_{\mathbf{offset}}(eventCounter + 1) \leftarrow \left[\ eventCounter + 1 \quad timeStamp \quad \delta\hat{x}^* \quad \delta\hat{y}^*\ \right]^T$

$\mathbf{E}^{(\mathbf{G})}_{\mathbf{offset}} \leftarrow \mathbf{E}^{(\mathbf{G})}_{\mathbf{offset}} + \mathbf{E}^{(\mathbf{G})}_{\mathbf{offset}}(eventCounter + 1)$

% Output selection under GPS rejection

$\widehat{\mathbf{S}}(M \cdot g + M) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{V})}(M \cdot g + M)$

else

% Output selection under GPS acceptance

$\widehat{\mathbf{S}}(M \cdot g + M) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{GV})}(g + 1)$

% Implementing GPS pin

$\widehat{\mathbf{S}}^{(\mathbf{V})}(M \cdot g + M) \leftarrow \widehat{\mathbf{S}}^{(\mathbf{GV})}(g + 1)$

end if

% Resetting offset detector

$eventCounter \leftarrow (eventCounter + 1)$

$offsetFlagX \leftarrow false$

$offsetFlagY \leftarrow false$

$\delta\hat{x}^* \leftarrow 0$

$\delta\hat{y}^* \leftarrow 0$

}

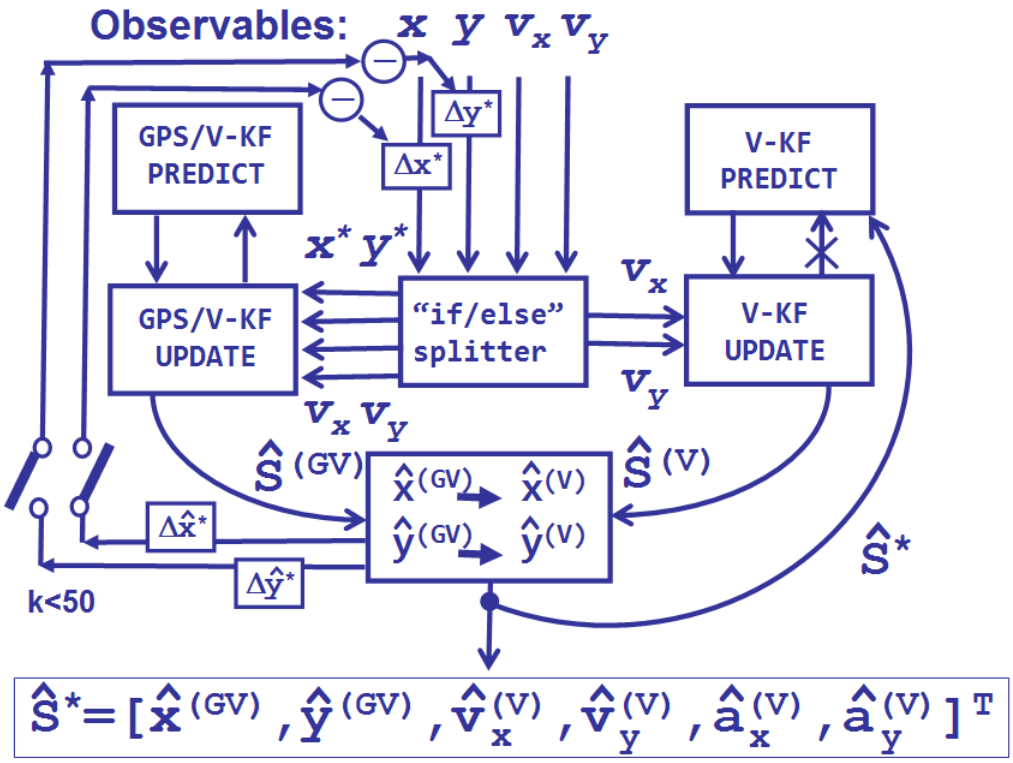


Figure 3.7: Architecture for GPS offset occurrence detection and compensation. The respective code structure for implementation is presented as Algorithm 6 on page 52 supplemented with Algorithm 7 on page 53. Introduction of switches allows to separate training cycle of the KFs from KFs' operation under GPS offset compensation, thus achieving beneficial reduction of interference with the KFs training cycle.

Chapter 4

Simulations and Performance Analysis

The simulation experiments herewith are designed to verify the efficiency of compositional KFs in basic operations over data streams in IoT. These operations include (a) filling in intermittent data,(b) recovering and/or replacing of the lost or missing data, (c) detecting error events and errors elimination and/or correction. The architectures proposed in Chapter 3 require verification, which is conducted herewith via simulation. The results are presented below. As a quantitative parameter of accuracy for variables representation, a root mean square error is used, abbreviated RMSE. The notation for RMSE also includes the measured parameter in brackets, so that RMSE(x) denotes root mean square error for coordinate x as a parameter. Similarly, RMSE(y), RMSE(v_x), and RMSE(v_y) are root mean square errors for coordinate y, and V-components, v_x and v_y.

4.1 Model Formulation

Let us define the model, suitable to test the effectiveness of our compositional KF approach for the chosen scenarios in IoT. This model would allow us to generate data streams, that are then contextually grouped and processed by specialized interacting KF units.

As it has already been mentioned in Chapter 3, here we choose the case of navigational data streams for the moving vehicle. This is because, the relationships for such streams are well understood, and these streams are essential part in the fleet management of IoT systems. Hence, analysis of operational principles of compositional KF becomes more clear. Moreover, the principles demonstrated for navigational data streams may further be explored for application in other types of IoT systems.

For the vehicle to navigate, we have chosen the trajectory shown in Figure 4.1 (page 56). As seen there, the trajectory starts and ends at the same point. Also, the trajectory is symmetrical with regard to its direction parallel to the $\vec{x}$ axis. The trajectory starts as a segment of straight line followed by a sequence of round turns and straight line segments. All the turns are smoothly rounded and there are no sharp corners on the trajectory, which allows for the smooth passage of the vehicle through this pathway. This serves to minimize the acceleration values acquired by the vehicle at points of changes in the driving conditions, such as exiting from the straight line segment of the trajectory and entering the turn, or vice versa.

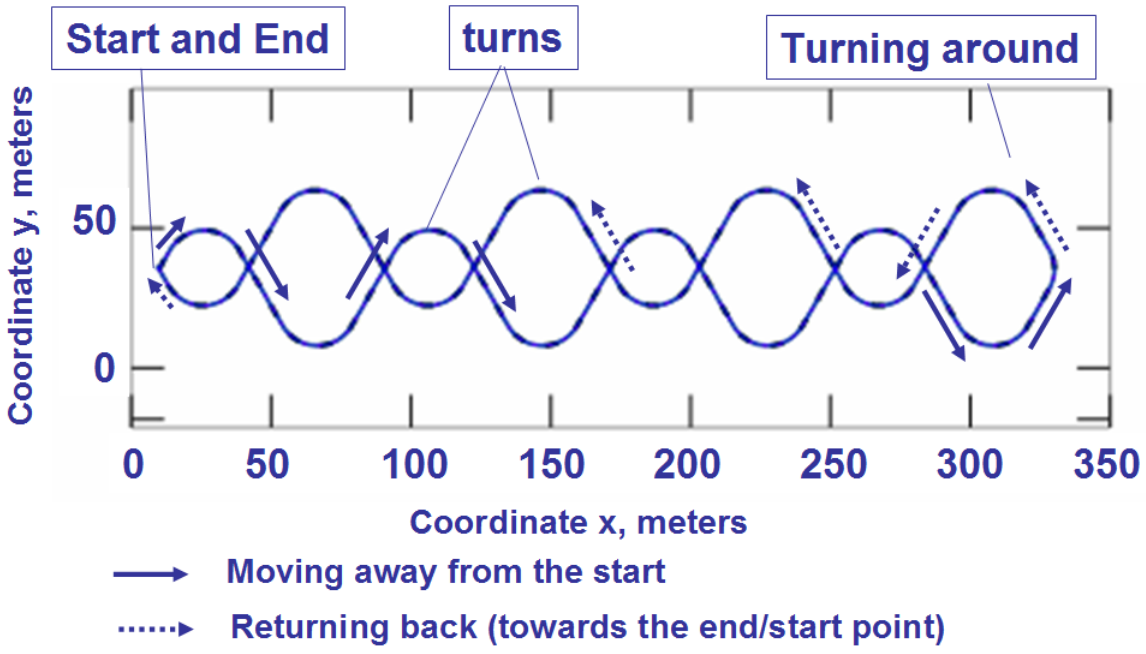


Figure 4.1: Two-dimensional trajectory for the test of car navigation. (Coordinate x is retrieved from GPS latitude signal, and coordinate y from GPS longitude signal).

The start point coordinate may be conveniently shifted away from the coordinate origin

to keep operating values of (x, y) coordinates in a positive quadrant. This, however, is not mandatory and has no effect on simulation results as all equations in the model allow any shift of the coordinate origin. In the example illustrated in Figure 4.1, the start point is (10, 26) (in meters) and the quantization step is 1 m long. The trajectory consists of 975 steps, thus being 975 m long, and is represented by N=976 points, with start and end points to be considered separate, although overlapping in the ideal case. The coordinates x and y are assumed to be retrievable from the values of latitude and longitude, supplied by the GPS. The radius of curvature for the turns everywhere is $r = (N-1)/(20\pi) = 15.52$ m and the turns are 29 m long (i.e. 29 steps in simulation) except for the turn-around circle, where the biggest value x is reached in the middle of a turn, which is 15 m long. Straight line segments are mostly 32 m long, except for the starting and ending segments, both of which are 8 m long, and segments entering and exiting the most remotely from the start point turn, both of which are 16 m long.

For convenience of description, let us assume that the coordinate system is aligned with the mapping net, so that North (N) is on the top, South (S) is at the bottom, West (W) is on the left hand side and East (E) is on the right. With this notation, the vehicle starts moving to the NE at the angle $\pi/3$ to x-axis, proceeds forward and then turns SE with angle $5\pi/3$ to x-axis, again proceeds forward and turns back to NE, repeating the move to make 8 turns in total before entering turn-around loop. Reverse trip symmetrically mirrors the forward trip, returning the vehicle back to the initial trip origin. We consider the vehicle to be moving at a constant speed of 10 m/sec, which is 36 km/hour, so that a full trip takes 975 seconds, or 16.25 minutes.

4.1.1 Model's Limitations

From the above model description, the following assumptions and simplifications adopted within the model as well as limitations imposed by the model's applicability become apparent.

Specifically, for the movement trajectory, the GPS precision effect is shown in Figure 4.2 for various measurement noise levels. The precision reflects the consistency, or in other words, the relative closeness, of the consecutive measurements to each other. For the GPS signal this would mean the consistency of the perceived trajectory. A zoomed in view of the trajectory for two different noise levels is seen in Figure 4.3.

For this model's trajectory, the turns are chosen to be smooth and the quantization for the turning curvature is set to be 4^o per meter (i.e. per step). Thus, the sharp turns which require a quantization under 4^o per meter are excluded from consideration.

The starting speed of the vehicle is not zero, but rather the max value upfront, which is 36 km/h (or 10 m/sec so that 1 step of 1 m takes 0.1 sec to complete). The acceleration effect takes place at the turns when velocity components v_x and v_y experience changes.

Additionally, the simplification of the model is that the length of the vehicle is neglected, which is regarded as reasonable approximation because we assign navigational characteristics to the vehicle's center of gravity.

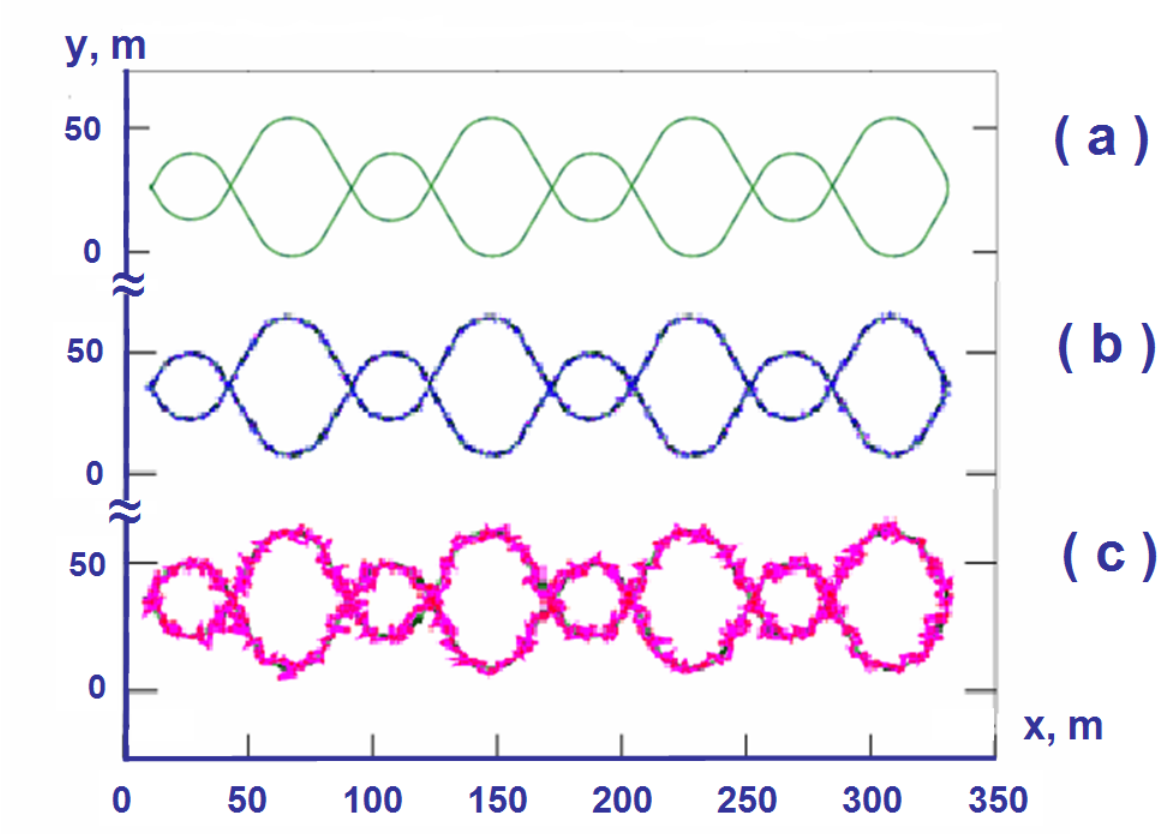


Figure 4.2: GPS view of 2D trajectory at various noise level: (a) $R = 0$, i.e. without measurement noise; (b) $R_{ii} = 0.4\ m^2$; (c) $R_{ii} = 4\ m^2$, $i = 1, 2$.

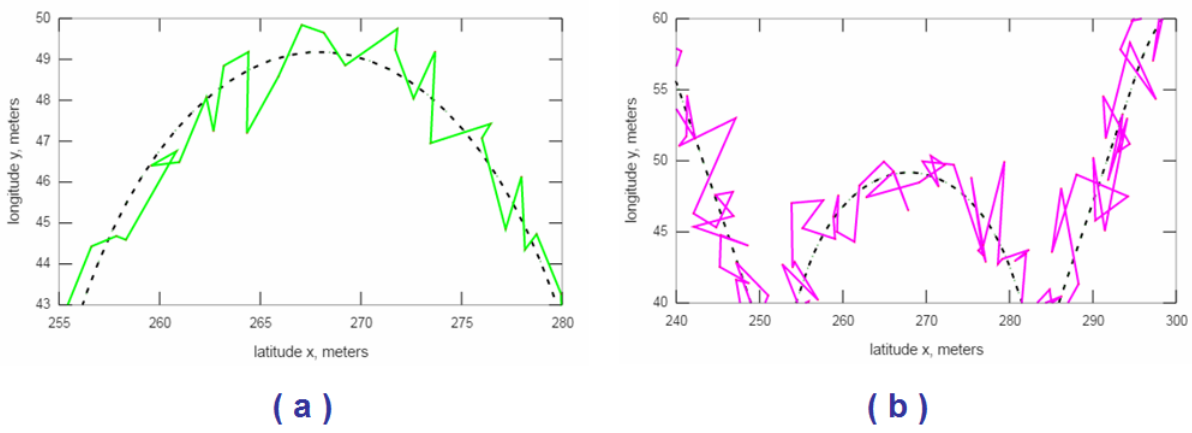


Figure 4.3: Details of GPS view of 2D trajectory at various noise level: (a) $R_{ii} = 0.4\ m^2$; (b) $R_{ii} = 4\ m^2$, $i = 1, 2$.

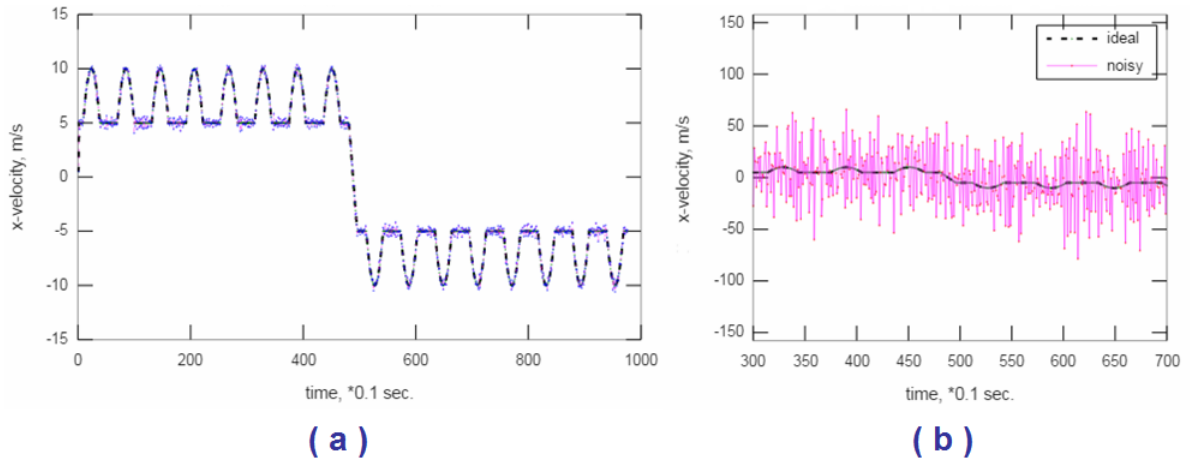


Figure 4.4: v_x evolution at various noise level: (a) R=0.0004 m^2; (b) R=4 m^2.

4.2 Imputation of Intermittent Data

In our model we adopt 10 Hz to be a maximum frequency for any sensor, but also allow operation at lower frequencies, such as 1 Hz, which is widespread in practice for GPS sensors. Therefore, if we monitor the GPS supplied coordinates at 1 Hz, but at the same time have variables v_x and v_y at 10 Hz, then the task of upgrading rate of GPS data flow to 10 Hz requires filling in the missing GPS data between sequentially coming at 1 Hz points. This is becomes the motion equation includes both, the coordinates as well as velocity components, hence both are needed at the highest rate to restore the trajectory as navigational output of the KF's observation.

When filling in the intermittent data, we shall keep in mind, for comparison purposes, the situation when all the data are initially present and there is no need for filling in any data. For our model, an example of this would be when all data streams operate at the highest adopted frequency of 10 Hz. Such a situation is depicted in Figure 4.5, where the cases for two noise levels are presented — lowest positive modelled measurement noise level, at which $R_{11} = R_{22} = 0.0004m^2$, and highest modelled measurement noise with $R_{11} = R_{22} = 4m^2$. The Kalman filter here combines data at 10 Hz rate from the GPS for x and y coordinates, as well as the speedometer and the orientation meter merged to yield v_x and v_y (also at 10 Hz rate).

4.2.1 Intermittent Data Imputation with Prediction Cycle

First, we test the efficiency of the intrinsic KF's ability to fill in intermittent data in predict-only mode, i.e. the mode when the KF's predict cycle is used at a higher frequency than the actual data are coming in.

The disadvantages of this approach can be seen in Figure 4.6, which presents the case of the conversion of a 1 Hz GPS stream of y-coordinates into a 10 Hz stream of the same

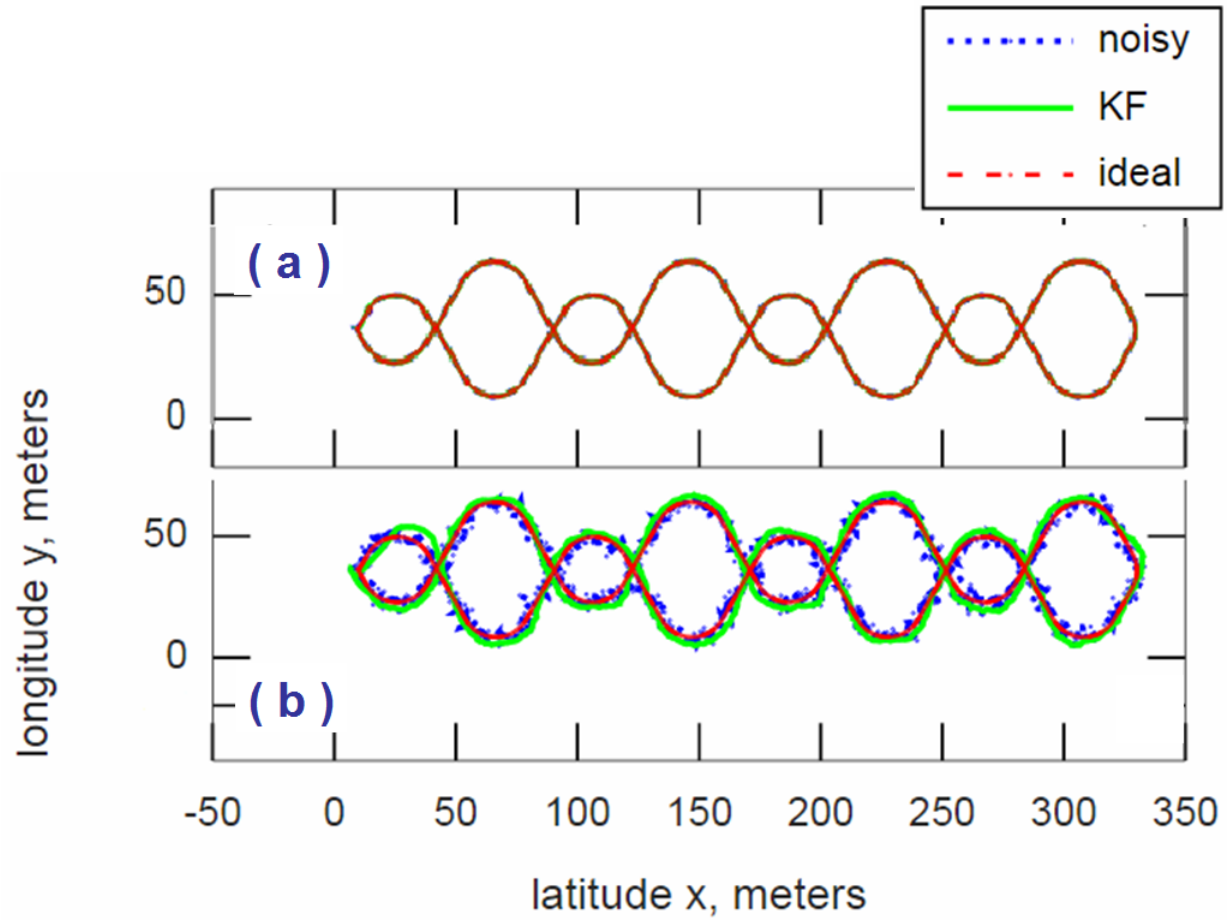


Figure 4.5: Performance of combined GPS/V KF (10 Hz) at various noise level: (a) R=0.0004 m^2; (b) R=4 m^2.

via filling in of periodic intermittent data. Here the KF is placing 9 intermediate points between initially given ones at 1 Hz rate, so that 1 point at each second is supplemented with the additional 9 points to produce the stream of 10 points per second of y-coordinates, thus yielding a 10 Hz data stream.

Overall the deficiency of this filling in approach is seen with regard to the accuracy provided . Despite this deficiency in accuracy, it may still be considered as a plausible approach under the absence of other alternatives. For the navigational streams, however, there is possibility of taking advantage of the redundancy of the related sensors, which brings in the conceptual possibility of filling in intermittent data from redundant sensory sources. As an example of this, the next subsection presents the performance of the V-KF module.

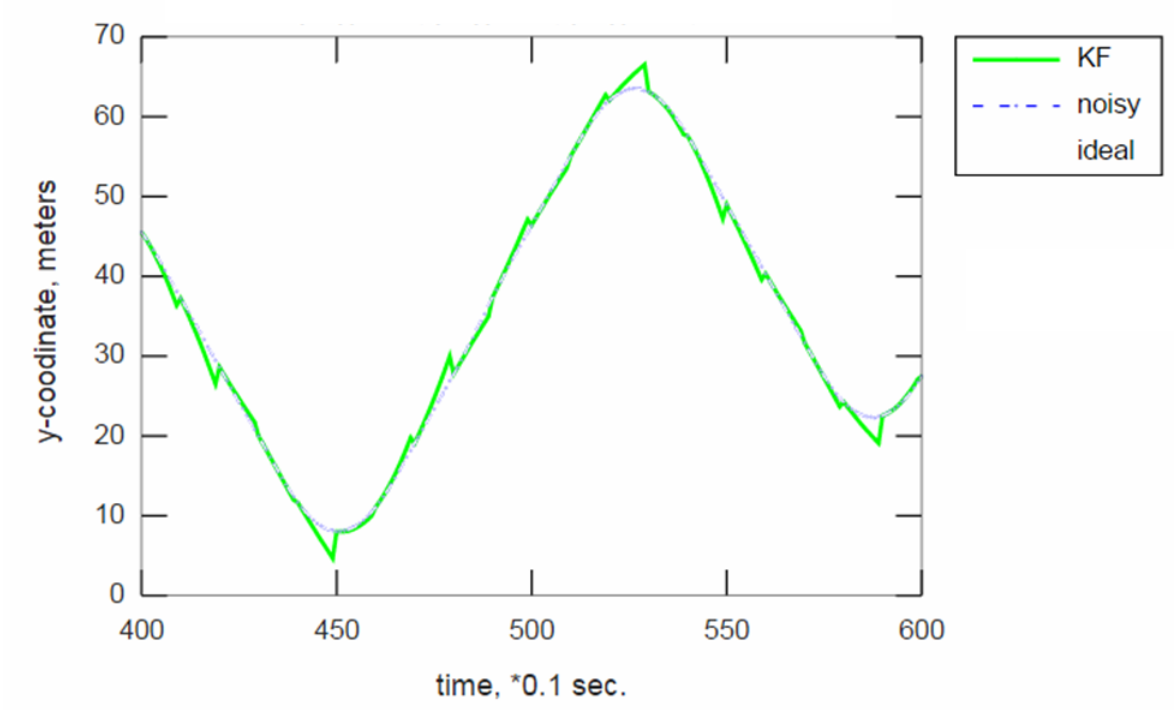


Figure 4.6: The y-stream: upgrading 1 Hz GPS y-data rate to 10 Hz by fillng in missing data with KF in predict-only mode under very small noise condition (R=0.0004 m^2), which yields RMSE(y)=2.9. (The "ideal"line would be hidden under the "noisy"one and thus not shown).

4.2.2 Intermittent Data in V-Mode

The practical implementation of employing the V-mode of the KF's operation is presented in Figure 4.9 (zooming into details in Figure 4.10). There, the V-KF (10 Hz), i.e. KF in V-only mode with no GPS signal all the way along, even at a very high noise level of $R_{33} = R_{44} = 800m^2/sec^2$ for v_x and v_y related components of covariance matrix $\mathbf{R}$ still manages to maintain the overall trajectory shape at the cost of the phase shift acquired within the training cycle, yielding RMSE(x)=31; RMSE(v_x)=2.7; RMSE(y)=14; RMSE(v_y)=7.1. Note here that for matrix $\mathbf{R}$ of our model a noise level with $R_{33} = R_{44} = 800m^2/sec^2$ for v_x and v_y components is equivalent to noise level with $R_{11} = R_{22} = 4m$ in coordinates x and y. The significantly lower values of the RMSE for velocity components as compared to the coordinate ones is reflecting that the accuracy here is achieved in the velocity domain at the expense of that in the coordinate domain, where the trajectory is perceived to be shifted with regard to its actual position.

Figure 4.9 shows the addition of the acquired phase shift values $\Delta x_0, \Delta y_0$ to the actual values of the coordinates x_{actual}, y_{actual}. This observation is in line with the relation described by equation 3.16 on page 38, thus supporting validity of our conceptual description of the KF's V-mode operation given in subsection 3.4.2 of Chapter 3. This justifies further verification of the formulated earlier concept of GPS-pins for V-KF as well as im-

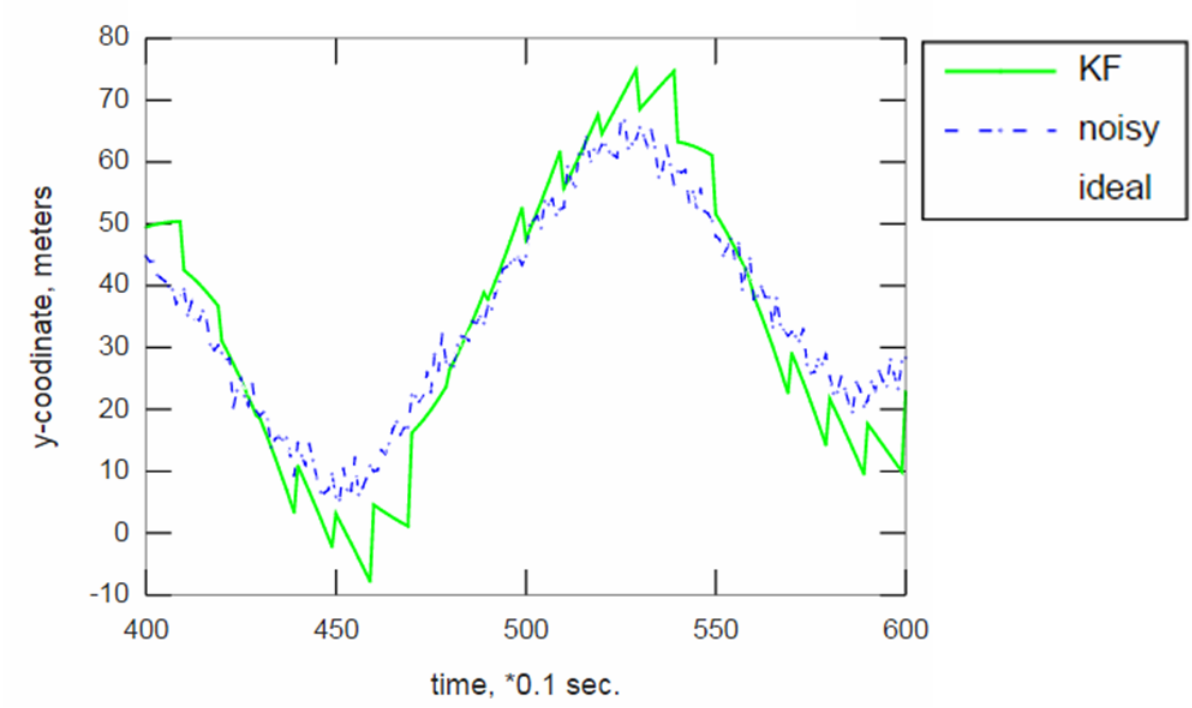


Figure 4.7: The y-stream: upgrading 1 Hz GPS y-data rate to 10 Hz by fillng in missing data with KF in predict-only mode under high noise condition (R=4 m^2), which worsens accuracy by yielding RMSE(y)=7.8. (The "ideal"line would be hidden under the "noisy"one and is omitted herewith).

plementation of the architecture proposed for this purpose and shown in Figure 3.5 on page 40. Therefore we proceed with more detailed consideration of the GPS pin-assisted architecture and its operation in the next subsection 4.2.3.

4.2.3 GPS Pins for V-Mode

Now we attempt extending the accuracy provided to v-components, v_x and v_y, by V-KF to coordinate components, x and y, via GPS pinning as described in subsection 3.4.3 of Chapter 3. The example of GPS pins implementation for V-KF is presented in Figure 4.11 for moderate measurement noise level for V-domain with v-related diagonal components of covariance matrix $\mathbf{R}$ being $R_{v_x} = R_{v_y}$ =8 m^2/sec^2. This noise level is equivalent to coordinates domain noise covariance components being 0.04 m^2, which is rated herewith as moderate noise level too. It is evident, that some phase shift is again acquired at the initial stage of the travel, when the V-KF undergo V-mode training. However, after training stage, at the very first pinning point the acquired so far phase shift gets compensated and the desired trajectory is followed quite closely by the KF at V-mode and the remaining pinning points are seen as practically redundant. This means that the V-KF with some rare pinning points offers excellent filling in capability of intermittent GPS-based coordinates data. Under such circumstances, it would be advantageous to keep V-KF as a separate

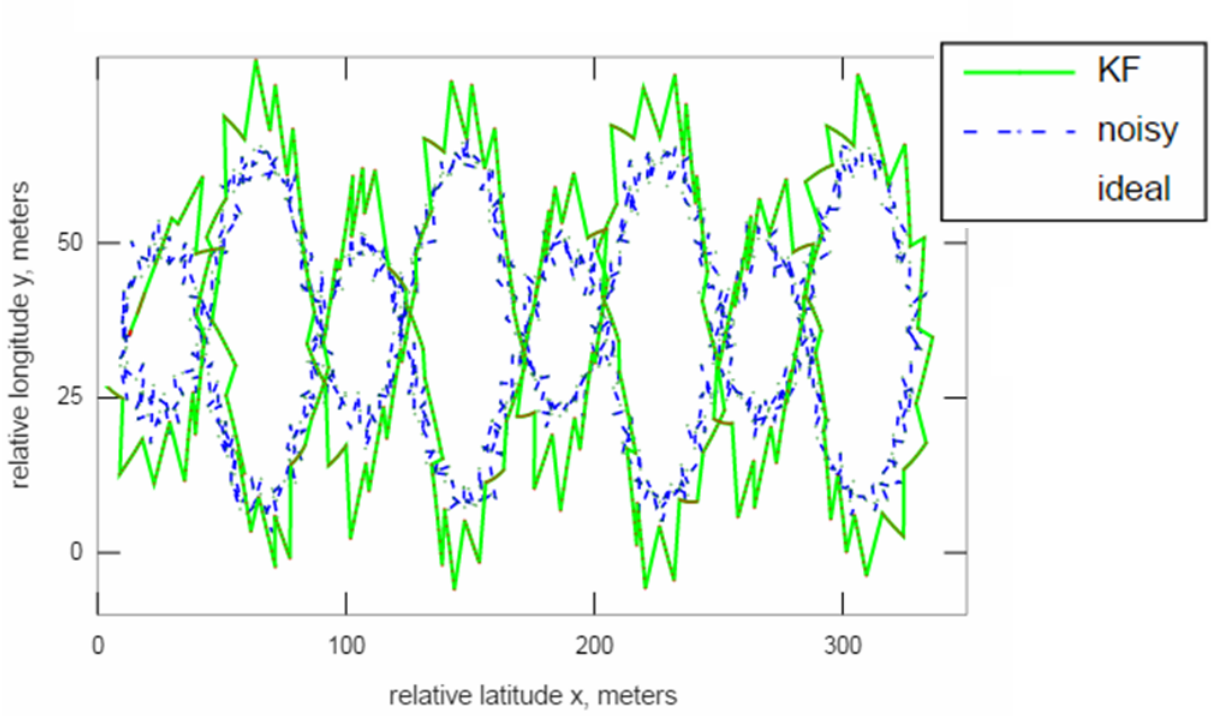


Figure 4.8: Upgrading 1 Hz GPS data rate to 10 Hz by fillng in missing data for trajectory with KF in predict-only mode at high noise level (R=4 m^2), which yields RMSE(x)=2.8 and RMSE(y)=7.5. (The "ideal" line would be hidden under the "noisy" one and is omitted herewith).

specialized unit in our compositional KF network for the task of intermittent data filling in. It is, however, important to note here, that because at pin points the excessive trust is exercised towards the pin source, the critical parameter for the quality of the pins becomes its measurement noise variance. Figure 4.12 demonstrates that noise increase of the GPS pin signal to the level of $R_x = R_y = 4\ m^2$ leads to random shifts of the trajectory at pin-points. The shape of the trajectory becomes piece-wise regular. In this case, between the pinning points the V-KF output trajectory is closely following the shape of the original, but is phase shifted by the value of the noise component at the moment of the last pin. It is therefore preferable that the noise level of the pins does not exceed that of the equivalent noise level of the v-data streams, as is the case for Figure 4.11.

If the noise level for both V-KF and GPS pin sequence is high, but the condition of the pin's noise not to exceed that of the V-KF holds, then the trajectory piece-wise inconsistency does not really actualize itself. In this case, the phase shift is compensated overall and the trajectory is followed within the noise limits, transforming the trajectory of Figure 4.9 into that of Figure 4.13 while keeping the trajectory coordinates error level low, yielding RMSE(x)=2.5, RMSE(y)=2.6 and somewhat higher errors for velocity components with RMSE(v_x)=3.8 and RMSE(v_y)=7.3.

This concludes consideration of intermittent data filling in with the help of the KF. As

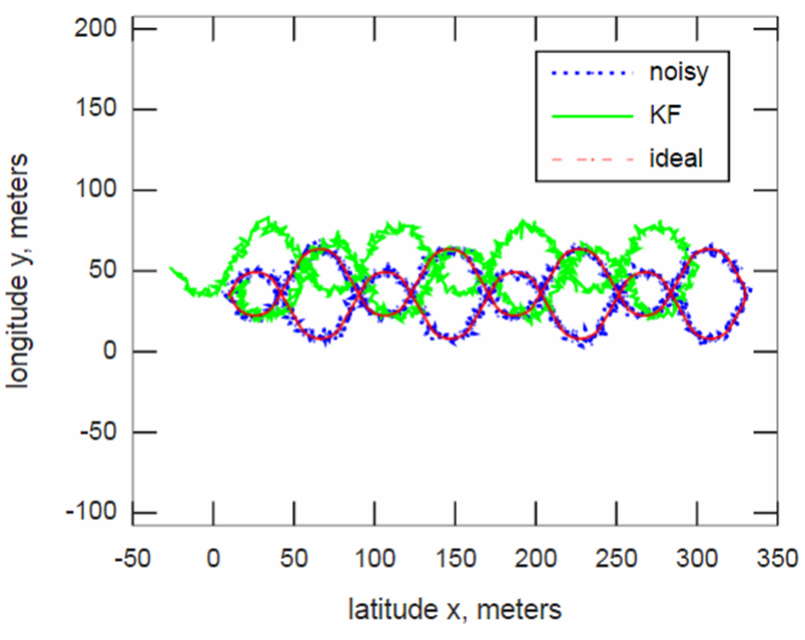


Figure 4.9: Phase shift for the V-KF (10 Hz) at high noise level (R=4 m^2) with no GPS signal, yielding RMSE(x)=31; RMSE(v_x)=2.7; RMSE(y)=14; RMSE(v_y)=7.1;

we have seen above, the redundancy of the data streams can be used to compensate for insufficient data. In particular, it allows to fill in the intermittent data even in cases of extended intervals. Further extension of this ability of the KF will be demonstrated in the next section 4.3 in instances of correcting erroneous data as well as recovering lost data.

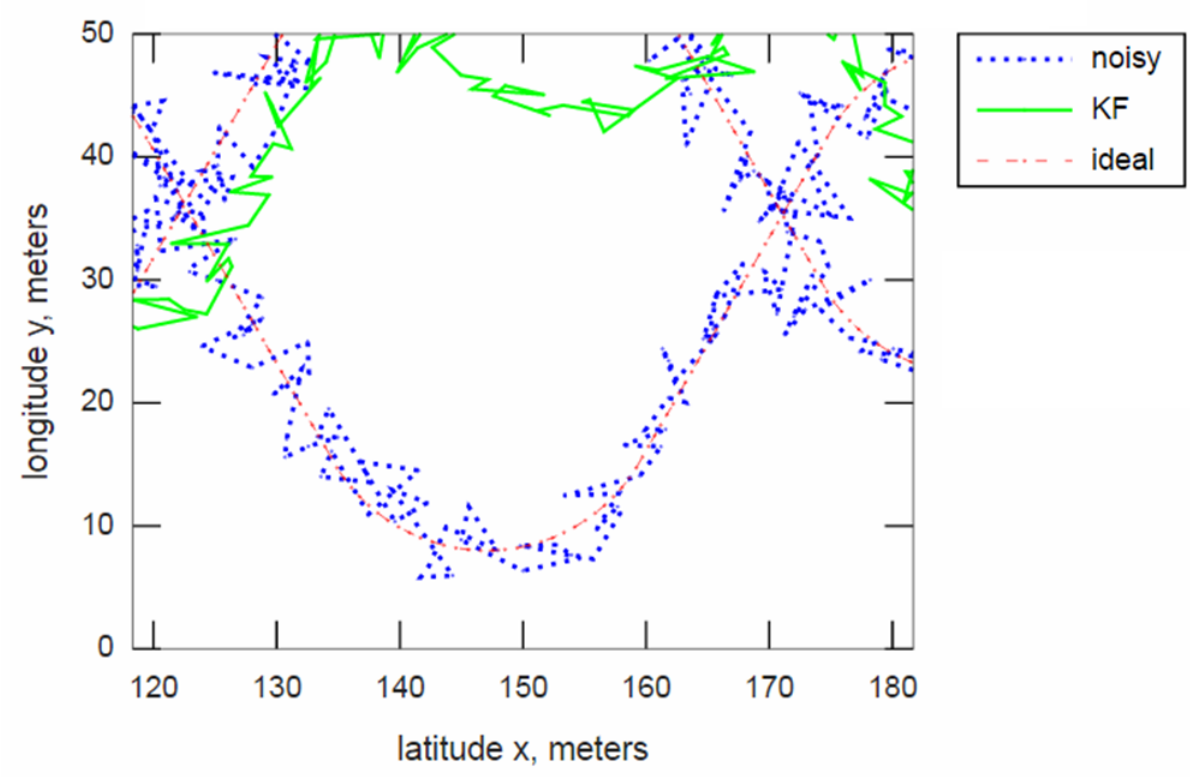


Figure 4.10: Details of V-KF (10 Hz) handling high noise (equivalent to R=4 m^2 in coordinate domain) with no GPS assistance. The yield is RMSE(x)=31; RMSE(v_x)=2.7; RMSE(y)=14; RMSE(v_y)=7.1.

4.3 Missing Data Recovery

As an example of missing data recovery the scenario of GPS signal outage is considered in this section. Relevant architectures are gathered in section 3.5 of Chapter 3.

4.3.1 GPS Outage Handling

Testing of the architecture of Figure 3.6 operating according to Algorithm 4 on page 46 has been implemented for the case when V-KF is turned off initially and is activated only when GPS outage occurs. The result is shown in Figure 4.14, where at the starting point A, the GPS/V-KF begins its training with incoming rate of 10 Hz of GPS and velocity components streams. The GPS outage occurs at the step k=120 of the simulation, i.e. at the 12-th second of the trip, marked as point B there. At this point the V-KF is activated and the output switch reconnects to position 0, which allows us to observe V-KF state vector estimates at the overall output of the scheme (case 0 in Figure 4.14). At that moment the V-KF starts the training stage, which is in V-mode and therefore aims to minimize the error in predicting the values of the velocity components v_x and v_y. The V-KF here

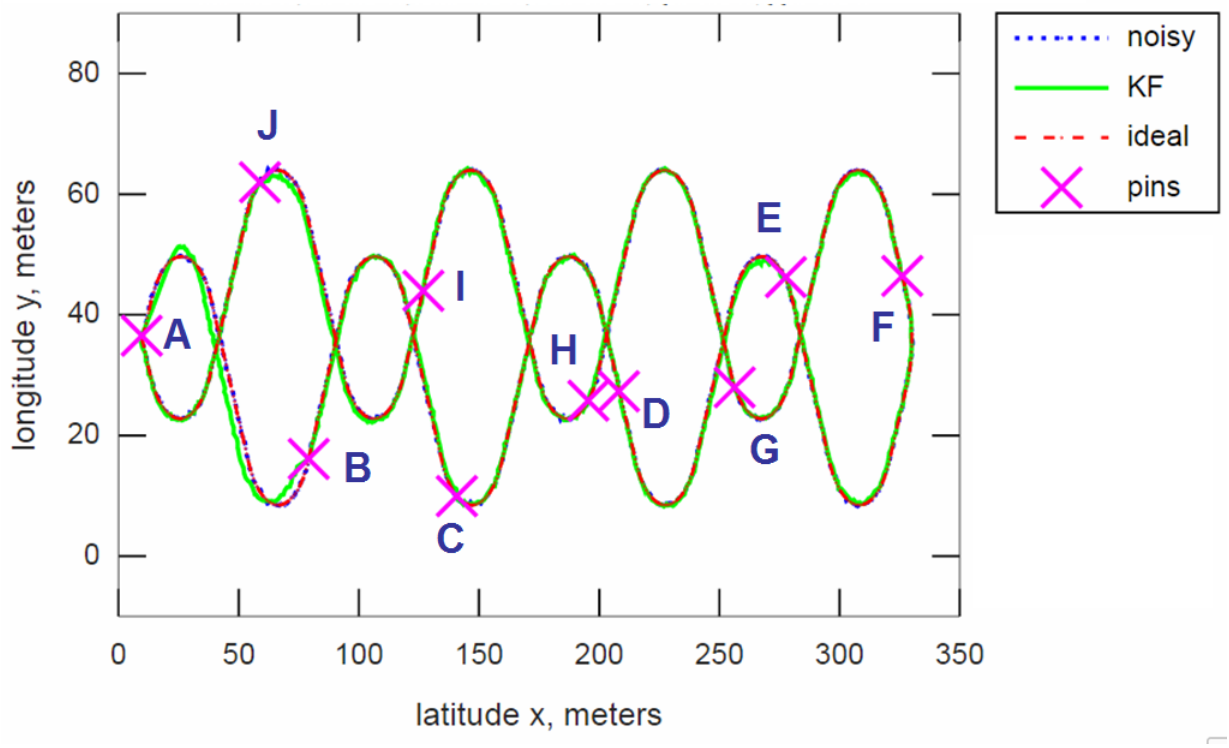


Figure 4.11: GPS pins (at 0.1 Hz) assist KF in V-mode at moderate noise level of R=0.04 m^2 for both GPS-KF as well as V-KF.

is completely blind to the errors in coordinates x and y and does allow accumulation of the phase shift in the position $\mathbf{\Delta r_1}$. Such accumulation of the phase shift saturates and ends at about point C' in Figure 4.14, at which point V-KF becomes able to correctly follow the shape of the moving vehicle's trajectory . It allows us to estimate the length of the training period to be around 3 seconds long or within 30 meters of the travel at a given speed. As a result of training, the V-KF converges and follows the shape of the trajectory by holding the value of acquired phase shift $\mathbf{\Delta r_1}$. The overall accuracy of the trajectory approximation by the V-KF can be estimated via values of RSME(v_x)=1.719, RSME(v_y)=2.961, both of which are reasonably small, and where the relation of RSME(v_x) < RSME(v_y) reflects the fact that V-KF training caused a larger shift along the y-axis and therefore errors were mostly made for v_y estimates. The high error values for coordinates RSME(x)=12.179 and RSME(y)=40.223 reflect the amplitude of the accumulated phase shift —$\mathbf{\Delta r_1}$— and its direction (about 3:1 in favour of y-shift compared to x-shift).

Because the shape of the trajectory is mostly preserved by V-KF, the GPS outage causes only partial loss of information. In the case where the phase shift $\mathbf{\Delta r_1}$ can be recovered afterwards (i.e. after completion of the trip and in the post-processing stage), for example from positioning of the end point A' and its known relation to the initial start point A as shown in Figure 4.14, then most of the lost information becomes recoverable by reversing the phase shift $\mathbf{\Delta r_1}$. In this case, only the period of the V-KF retraining (from point B to about point C') contains most inaccuracies in the data. This is relatively small amount

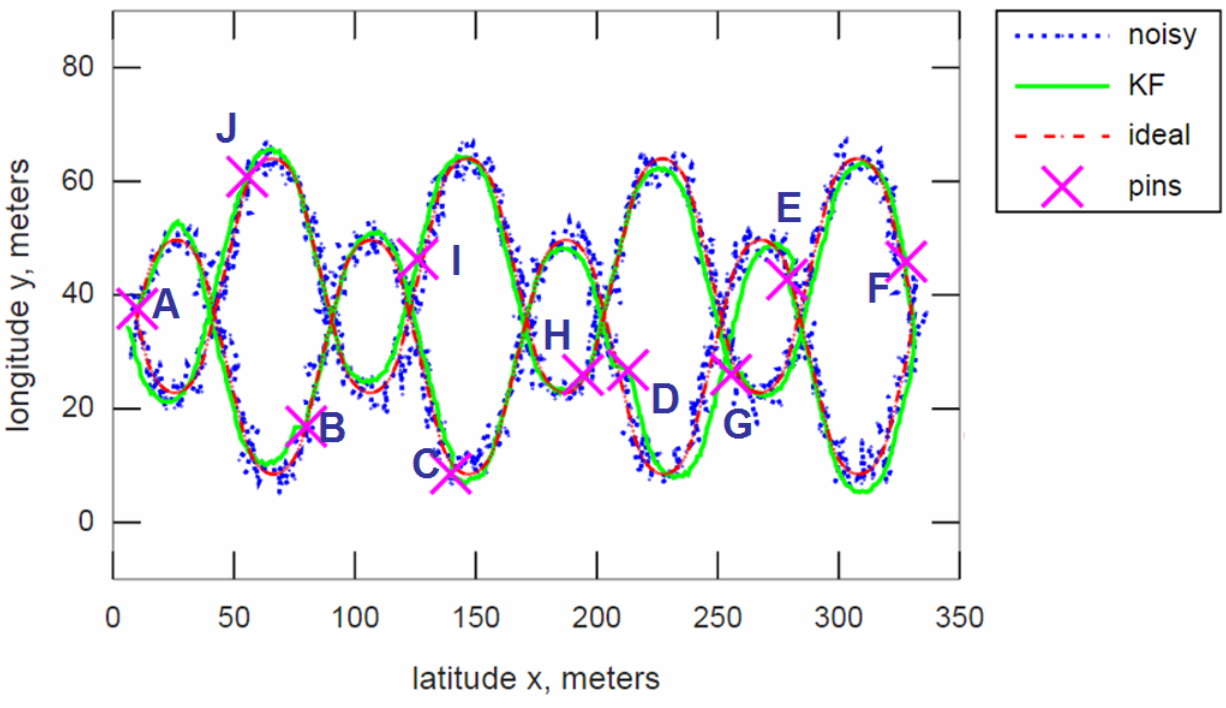


Figure 4.12: GPS pins (at 0.1 Hz) are noisy too at high noise of GPS-KF (R=4 m^2) at moderate noise level of V-KF (equivalent to R=0.04 m^2 in coordinate domain).

of the data and it may be further reduced via post-processing routines (interpolations, backward Kalman filtering, etc.). However, it would be desirable to have a real time version of the solution to the accuracy lost at the GPS outage. Otherwise, some tasks fall out of reach for the system, for example entering the tunnel on automatic control mode with parameters as in Figure 4.14 would cause failure or accident after the B-point.

The solution to the problem can be found in the fact that we need to eliminate the training phase for the V-KF after GPS outage. To solve this, we can move the training cycle of the V-KF to the time period preceding the GPS outage event, i.e. to the part A-to-B of the trajectory. In practice, we can use velocity components v_x and v_y streams for idle pre-training of the V-KF at the beginning of the trip, i.e. from the point A in Figure 4.14. This would require the modification of the code structure from Algorithm 4 on page 46 to the code of Algorithm 7 on page 47. Newly modified code would allow more than sufficient time to train the V-KF in idle fashion, i.e. discarding its output $\widehat{\mathbf{S}}^{(\mathbf{V})}$, in order to benefit from immediate take over by $\widehat{\mathbf{S}}$ in transition from $\widehat{\mathbf{S}}^{(\mathbf{GV})}$ to $\widehat{\mathbf{S}}^{(\mathbf{V})}$ by pre-trained V-KF when the output switch in Figure 3.6 shifts from position 1 to position 0 due to a detected GPS outage event.

Verification of this fact for the scenario in Figure 4.14 confirms that idle pre-training does provide immediate and accurate control transfer from GPS/V-KF to V-KF, as Figure 4.15 on page 70 illustrates. Under such control transfer, entering the tunnel in automatic regime becomes much safer (as safe as with GPS signal inside the tunnel).

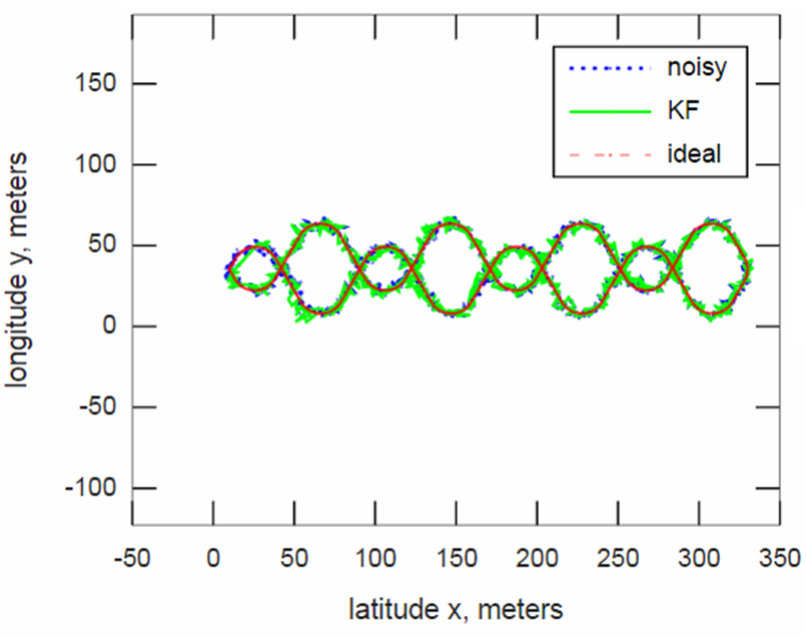


Figure 4.13: V-KF (10 Hz) with GPS-KF (1 Hz) at high noise level (equivalent to R=4 m^2 in coordinate domain) yielding RMSE(x)=2.5; RMSE(v_x)=3.8; RMSE(y)=2.6; RMSE(v_y)=7.3.

4.4 Errors Elimination Tests

4.4.1 GPS Offset Occurrences

The simulated example of the occurrence of the GPS offset is shown in Figure 4.16 for the case of GPS/V-KF and V-KF connected for GPS pinning as in Figure 3.5 with availability of GPS signal at 1 Hz and velocity components at 10 Hz. It is seen that the GPS pinning here is counter-productive in that it enforces GPS signal offset as a real signal. This is not the intended behaviour and as a next step we will consider how to actually detect the occurrence of the GPS offset as well as how to approach the task of its compensation.

4.4.2 GPS Offset Compensation

For the initial testing of the architecture shown in Figure 3.7 of page 54 and operating via Algorithm 5 on page 52 supplemented with Algorithm 6 on page 53, a sufficiently high threshold values have been chosen and its performance can be seen in Figure 4.17 on page 72.

In this simulation, GPS offset occurred on step 120 of the time count (which is 120 meters in the travel distance) and reversed on the step 420. In both cases, the offset event was detected and the noisy input signal represented by the blue dotted line in Figure 4.17 has been phase shifted accordingly to compensate the GPS-related offset. There is, however,

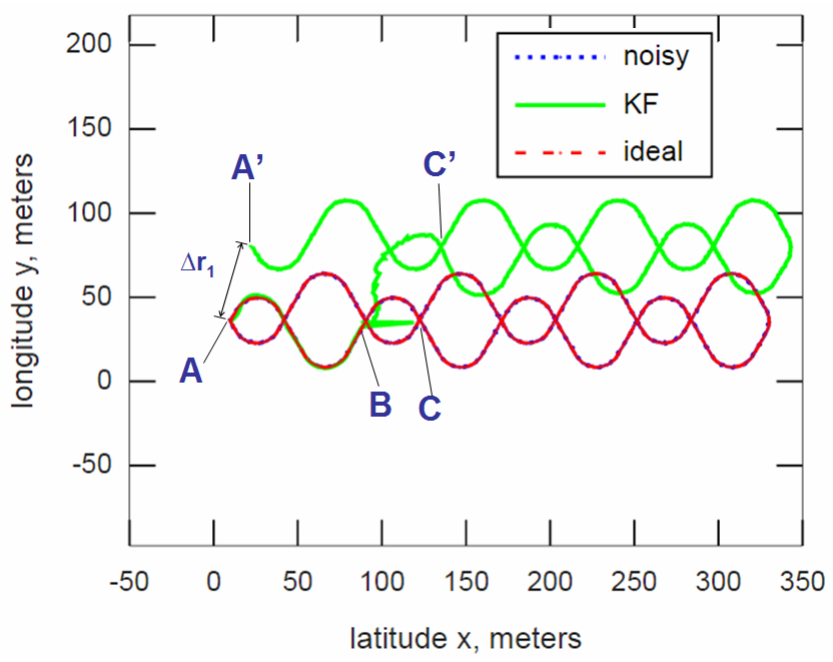


Figure 4.14: GPS outage causes retraining of KF in V-only mode (at moderate noise level, equivalent to R=0.04 m^2 in coordinate domain). Overall RSME(x)=12.179, RSME(y)=40.223, RSME(v_x)=1.719, RSME(v_y)=2.961. All sensors work at 10 Hz rate.

the issue of the initially acquired phase shift, which is seen in Figure 4.17 to be persistent through the whole travel length. This phase shift initially occurred during the training stage of the KFs. During the training period the parameters of the KF, such as Kalman Gain, undergo tuning to converge by minimizing predictive error. Such convergence can be detected as minimization of the state covariance matrix elements values. The relevant example is shown in Figure 4.18 on page 73, from which the strong changes of the matrix element $\mathbf{P}^{(\mathbf{GV})}[1,1]$ are seen until the 5-th second from the beginning of the trip (which is division 50 on the time axis). After that, the changes acquire a repeatable pattern slightly oscillating between upper and lower limits with the GPS signal incoming rate of 1 Hz.

As the variations of the predictive values are highest during the training period, it is logical to assume that the threshold of the GPS offset detection has been exceeded by variations of the KF parameters during the training and this event has been erroneously interpreted and then treated as a small GPS offset. The compensatory procedures followed, as enforced by the Algorithm 5 of page 52, thus resulting in the detection of false positives, i.e. false events of GPS offset occurrences, which in turn created false compensation at the input, and which persisted thereafter. Therefore, the task of the compensation of such initial shifts arises.

Here, we suggest a solution, which lies in avoiding the interference of the offset-compensatory mechanism with the training period of the KFs altogether. The training period in our modelled scenario occurs at the beginning of the trip, when the parameters of the KF only initialized and need to acquire the optimal value by means of, and as a result of, undergoing

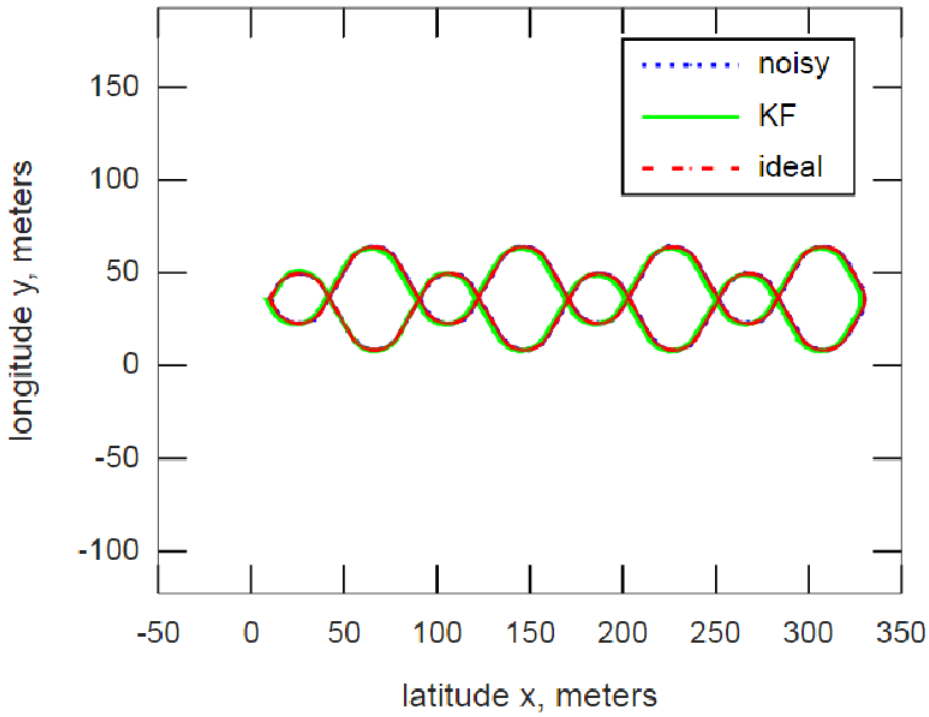


Figure 4.15: Pre-training of error covariance matrix $\mathbf{P}^{(\mathbf{V})}$ in idle mode of V-KF operation prior to event of GPS outage allows smooth transition of switching to V-mode at the occurrence of GPS outage. (Scenario of GPS outage event as in Figure 4.14 on page 69. Moderate noise level, R=0.04 m^2. Overall RSME(x)=1.581, RSME(y)=0.725, RSME(v_x)=1.738, RSME(v_y)=2.963. All sensors work at 10 Hz rate.)

the training. The training is actual exposure of the initialized KF to the data flows with representative characteristics and noise levels to ensure convergence of the KF by means of tuning the KF parameters to its optimal values for the given operational conditions. For this purpose, we suggest disconnecting the offset correcting mechanism for the duration of the training period of the KF. Here, the training may also include re-training due to significant changes in operating conditions. We therefore must introduce into suggested architecture shown in Figure 3.7 of page 54 switches for turning off the offset correcting mechanism by disconnecting the compensatory lines at the beginning of the trip for the duration of time certainly exceeding the duration of the KFs training. The two switches shown in the compensatory lines are turned off for initial 50 steps (see indication of $k < 50$ in Figure 3.7 on page 54) to allow training of the KFs to be completed. The length of training has always been under 30 steps, so 50 steps is a reasonable guess in an attempt to pass beyond the training completion point. When the compensatory lines are off, the KFs become vulnerable to the accidental occurrence of the GPS offset, which is therefore not allowed during the KFs training and which would need to be enforced as a precondition for the system to operate accurately.

The respective alteration of the code structure to reflect the design alteration resulting

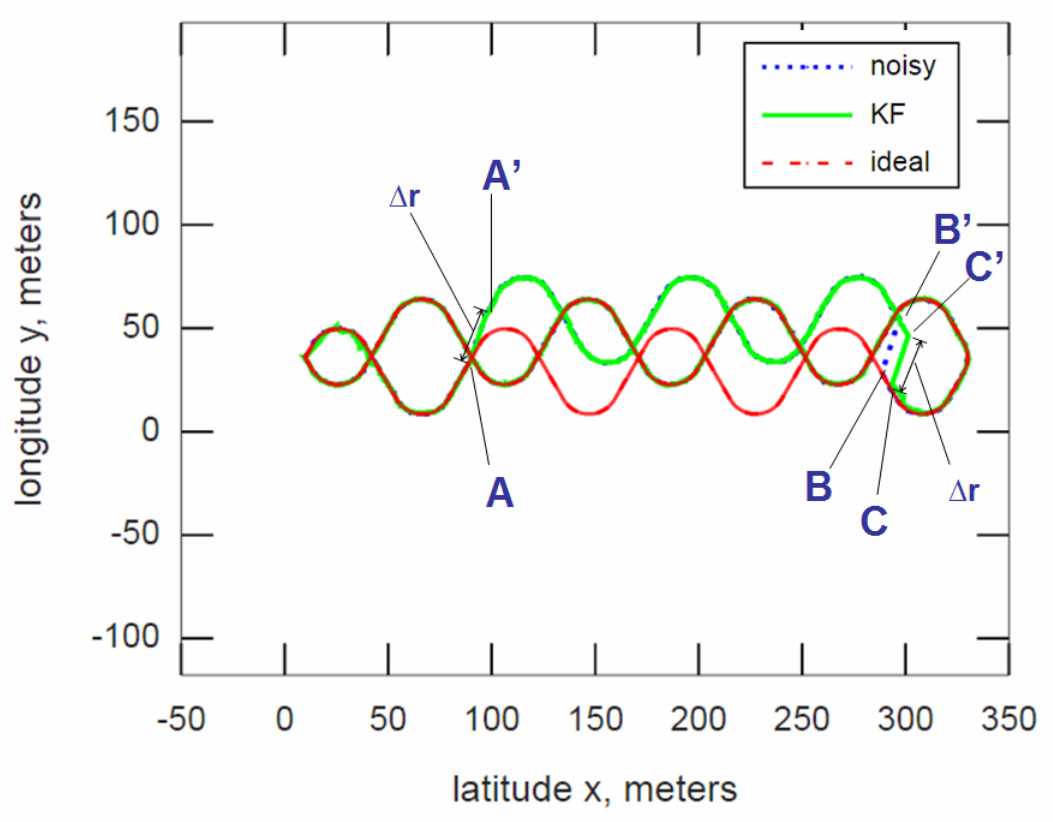


Figure 4.16: GPS offset occurrence for the combination of 10Hz V-KF and 1 Hz GPS-KF at moderate noise level of R=0.04 m^2 equivalence in coordinate domain for both. (Point A is at step 120 m with Δr=26.57 m, Δr_x=9 m, Δr_y=25 m; point B is at step 420 m with shift reversal of the same amplitude).

in the architecture of Figure 3.7 (page 54) consists of expanding the restrictive "if" condition in accessing GPSoffsetDetection() routine call of Algorithm 5 on page 52 with the addition of the clause "AND training phase is completed" in general case.

For our specific model, the general formulation of the restrictive additional clause "AND training phase is completed" would become "AND $k > 50$", meaning that GPS detection lines become connected after the 50-th step in simulation, i.e. after 5-th second from the moment of starting the trip. The result of testing the offset compensatory algorithm, with excluded interference during the training cycle, i.e. with the expanded restrictive additional clause "AND $k > 50$", is shown in Figure 4.19. Certain reduction of the undesirable initial phase shift as compared to that in Figure 4.17 can be seen, which indicates that our interpretation of its origin is in line with the simulation experiment.

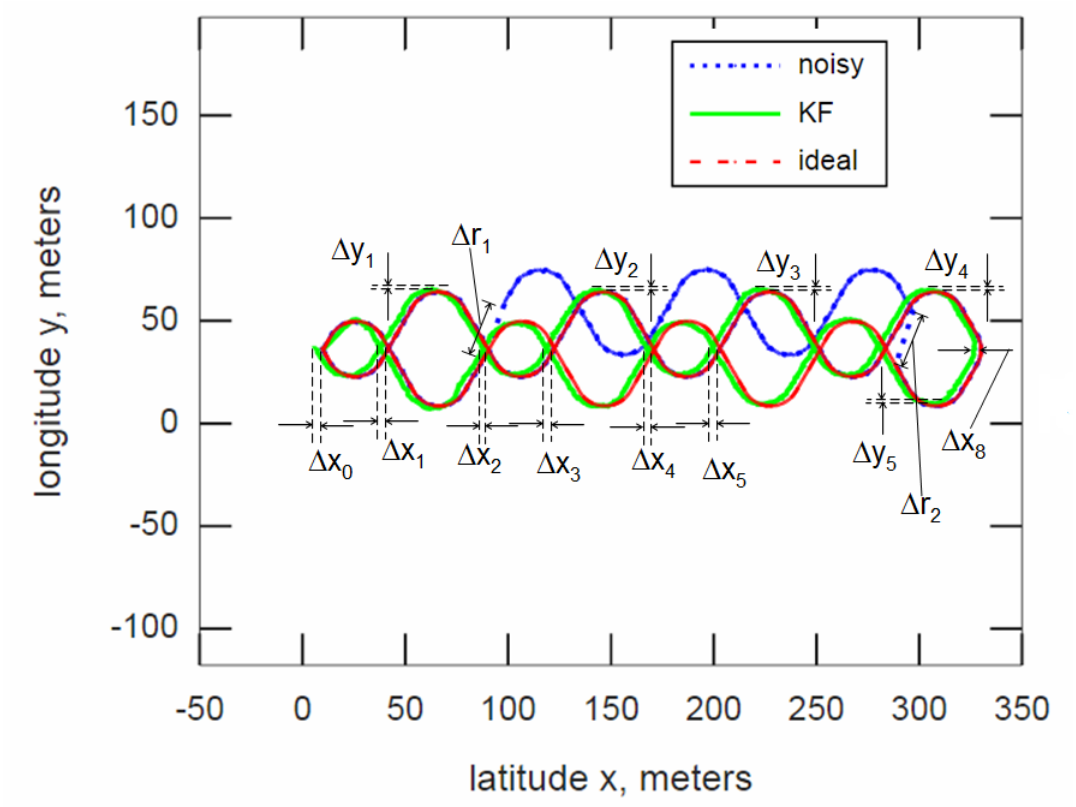


Figure 4.17: Compensation of the GPS offset while allowing interference with the KF's training cycle. (The parameters of the KFs, noise and GPS offset are the same as in Figure 4.16).

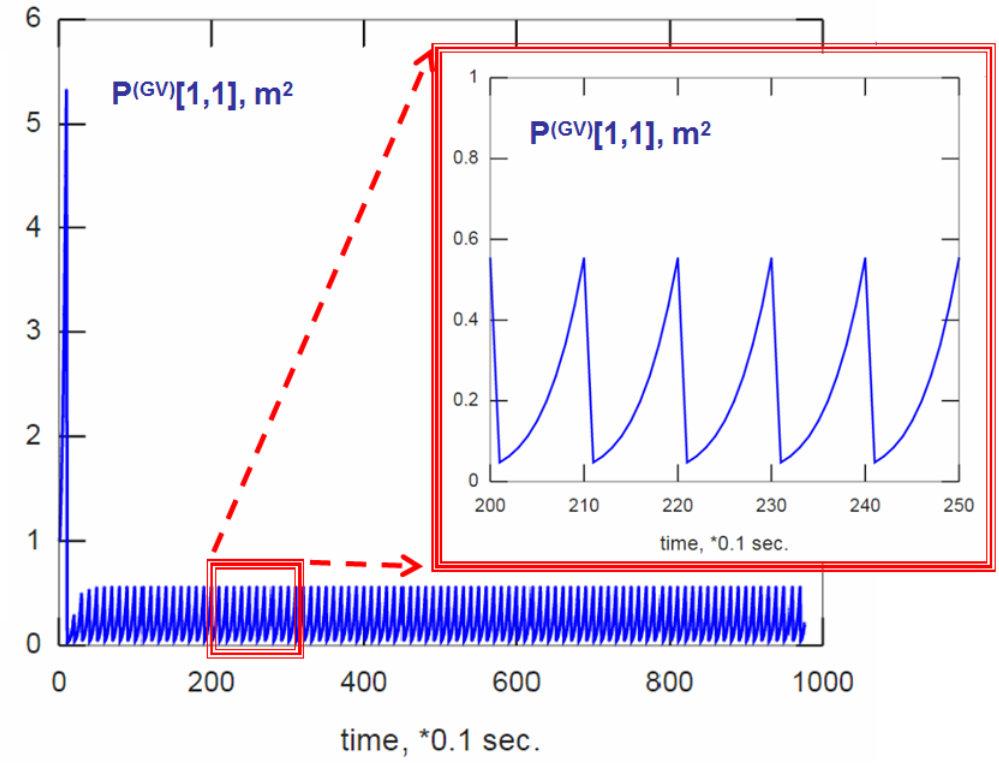


Figure 4.18: State covariance matrix $\mathbf{P}^{(\mathbf{GV})}$ elements evolution exemplified by its x-related element $\mathbf{P}^{(\mathbf{GV})}[1,1]$ in GPS/V-KF predict cycle at 10 Hz with GPS update at 1 Hz in implementation of Algorithm 5 on page 52 (The parameters of the KFs, noise and GPS offset are the same as in Figure 4.16).

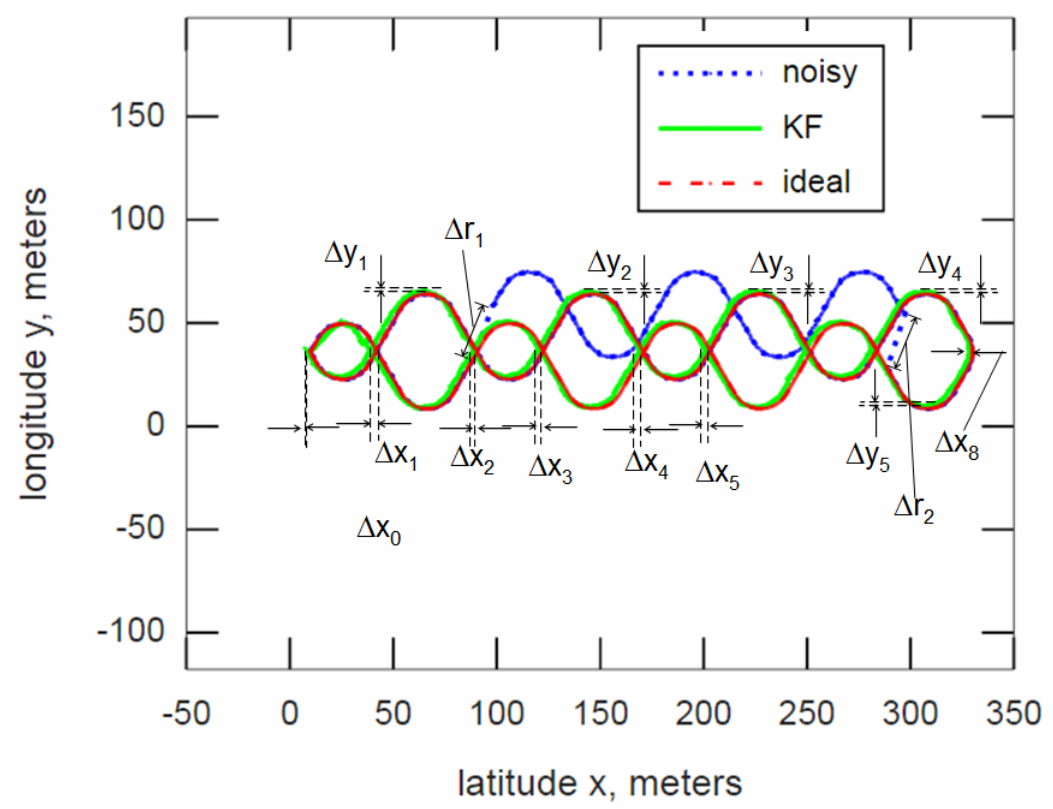


Figure 4.19: Compensation of the GPS offset involving algorithmic reduction of interference with the KFs training cycle. (The parameters of the KFs, noise and GPS offset are the same as in Figure 4.16).

Chapter 5

Conclusion and Future Work

To conclude, the presented thesis has explored the potential of the Kalman filter algorithm to benefit IoT systems with efficient real time pre-processing of data streams from the sensor networks. Pre-processing of the noisy navigational data streams of vehicles in two-dimensional motion, with coordinates from the GPS as well as velocity and acceleration components, is shown to allow efficient monitoring of the state of vehicles. We have demonstrated the ability of the Kalman filter algorithm to perform on-the-fly IoT-related pre-processing of the data, achieving noise reduction, filling in the intermittent data, lost or missing data recovery, error events detection and correction of detected errors. For these purposes a specialized architectural solution capable of performing the above tasks, called the compositional Kalman filter, is proposed. The functionalities of specific architectural implementations of compositional Kalman filters are verified through simulation experiments. Specific scenarios modelled by the simulations included equalization of the frequency of data flow for coordinate and velocity components, as well as GPS signal failures, such as GPS outage and GPS offset occurrences. The redundancy in the data streams, such as the one existing between coordinates, velocity and acceleration components has been employed to resolve the data processing problems in the above scenarios. The compositional Kalman filter architectures allow us to gather the data streams in IoT fleet management into mini-groups according to the contextual relevance of IoT and analyse them with Kalman filters with a minimized number of state variables involved. Demonstrated herewith are algorithmic solutions for interactions between different types of Kalman filters, acting as elementary building blocks of larger compositional architectures, capable of accomplishing the following IoT tasks:
1) data replacement and recovery for intermittent data imputation, which is demonstrated via data flow frequency upgrade, such as GPS signal upgrade from 1 Hz to 10 Hz;
2) missing data recovery, which is shown via GPS signal outage event detection and compensation;
3) outlier/error detection and correction, which is accomplished by GPS signal offset occurrence detection and compensation.
The conducted simulation experiments support the efficiency of the approach of building the composition of elementary Kalman filters into interacting networks for data correction and recovery, intermittent data imputation, missing or lost data recovery, and error events

detection and correction. Such interconnected networks of Kalman filters, or compositional Kalman filters, are proposed for further development as efficient pre-processing units for data streams originating from IoT system sensors.

Future work can include extension of the proposed architectures to the state of complete compatibility with real devices by removing currently imposed restrictions and assumptions made. A three-dimensional version with an extended set of navigational sensors, including gyroscope should be further explored. Non-linearity of the systems, excluded at this stage, may benefit more advanced versions of the architectures in question. The navigational data streams may be supplemented with data streams from the sensors observing the operation of the crucial parts of vehicles, such as engine and transmission. There is also a potential benefit in the optimization of conditions related to vehicles exploitation and maintenance, resulting in the eventual extension of vehicles lifespan.

References

[1] M Athans and PL Falb. *Optimal control*. McGraw-Hill, 1966.

[2] Michael Athans and David Kendrick. Control theory and economics: A survey, forecast, and speculations. *IEEE Transactions on Automatic Control*, 19(5):518–524, 1974.

[3] Peter J Brockwell and Richard A Davis. *Time series: theory and methods*. Springer Science & Business Media, 1991.

[4] Stef Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in r. *Journal of statistical software*, 45(3), 2011.

[5] Bradley P Carlin, Nicholas G Polson, and David S Stoffer. A monte carlo approach to nonnormal and nonlinear state-space modeling. *Journal of the American Statistical Association*, 87(418):493–500, 1992.

[6] Chris K Carter and Robert Kohn. On gibbs sampling for state space models. *Biometrika*, pages 541–553, 1994.

[7] WC Cave. Prediction theory for control systems. *Prediction Control Systems*, 2011.

[8] James Durbin and Siem Jan Koopman. A simple and efficient simulation smoother for state space time series analysis. *Biometrika*, pages 603–615, 2002.

[9] Sylvia Frühwirth-Schnatter. Data augmentation and dynamic linear models. *Journal of time series analysis*, 15(2):183–202, 1994.

[10] Unai Garciarena Hualde. An investigation of imputation methods for discrete databases and multi-variate time series. Master's thesis, University of the Basque Country, 2016.

[11] Arthur Gelb. *Applied optimal estimation*. MIT press, 1974.

[12] James Douglas Hamilton. *Time series analysis*, volume 2. 1994.

[13] A.C. Harvey. *Forecasting, Structural Time Series and the Kalman Filter*. Cambridge University Press, 1989.

[14] Andrew C Harvey and PHJ Todd. Forecasting economic time series with structural and box-jenkins models: A case study. *Journal of Business & Economic Statistics*, 1(4):299–307, 1983.

[15] James Honaker and Gary King. What to do about missing values in time-series cross-section data. *American Journal of Political Science*, 54(2):561–581, 2010.

[16] Philip K Hopke, Chuanhai Liu, and Donald B Rubin. Multiple imputation for multivariate data with missing and below-threshold measurements: Time-series concentrations of pollutants in the arctic. *Biometrics*, 57(1):22–33, 2001.

[17] Markus Hürzeler and Hans R Künsch. Approximating and maximising the likelihood for a general state-space model. In *Sequential Monte Carlo methods in practice*, pages 159–175. Springer, 2001.

[18] Joao Tovar Jalles. *Structural time series model and the Kalman filter: a concise review*. FEUNL Working Paper Series wp541, 2009.

[19] Richard H Jones. *Longitudinal data with serial correlation: a state-space approach*. CRC Press, 1993.

[20] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[21] Helio S Migon, Dani Gamerman, Hedibert F Lopes, and Marco AR Ferreira. Dynamic models. *Handbook of Statistics*, 25:553–588, 2005.

[22] Steffen Moritz, Alexis Sardá, Thomas Bartz-Beielstein, Martin Zaefferer, and Jörg Stork. Comparison of different methods for univariate time series imputation in r. *arXiv preprint arXiv:1510.03924*, 2015.

[23] Sohae Oh. *Multiple Imputation on Missing Values in Time Series Data*. PhD thesis, Duke University, 2015.

[24] DSG Pollock. Recursive estimation in econometrics. *Computational statistics & data analysis*, 44(1):37–75, 2003.

[25] Raquel Prado and Mike West. *Time series: modeling, computation, and inference*. CRC Press, 2010.

[26] Trivellore E Raghunathan. Combining information from multiple surveys for assessing health disparities. *Allgemeines Statistisches Archiv*, 90(4):515–526, 2006.

[27] Trivellore E Raghunathan, James M Lepkowski, John Van Hoewyk, and Peter Solenberger. A multivariate technique for multiply imputing missing values using a sequence of regression models. *Survey methodology*, 27(1):85–96, 2001.

[28] Jerome P Reiter. Simultaneous use of multiple imputation for missing data and disclosure limitation. *Survey Methodology*, 30(235):l242, 2004.

[29] Jerome P Reiter and Trivellore E Raghunathan. The multiple adaptations of multiple imputation. *Journal of the American Statistical Association*, 102(480):1462–1471, 2007.

[30] D.B. Rubin. Multiple imputation for nonresponse in surveys. new york: John wiley. 1987.

[31] Fred C Schweppe. *Uncertain dynamic systems*. Prentice Hall, 1973.

[32] Mike West and Jeff Harrison. *Bayesian forecasting and dynamic models*. Springer New York, 1997.