Institutionen för systemteknik
Department of Electrical Engineering

Examensarbete

Multiple Platform Bias Error Estimation

Examensarbete utfört i Reglerteknik
vid Tekniska högskolan i Linköping
av

Åsa Wiklund

LITH-ISY-EX-3501-2004

Linköping 2004

TEKNISKA HÖGSKOLAN
LINKÖPINGS UNIVERSITET

Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings tekniska högskola
Linköpings universitet
581 83 Linköping

Multiple Platform Bias Error Estimation

Examensarbete utfört i Reglerteknik
vid Tekniska högskolan i Linköping
av

Åsa Wiklund

LITH-ISY-EX-3501-2004

Handledare: **M.Sc. Gustaf Hendeby**
ISY, Linköpigs universitet
M.Sc. Pär Norberg
SAAB AB
Ph.D. Måns Mångård
SAAB AB

Examinator: **Prof. Fredrik Gustafsson**
ISY, Linköpigs universitet

Linköping, 26 January, 2004

Avdelning, Institution
Division, Department

Division of Automatic Control
Department of Electrical Engineering
Linköpings universitet
S-581 83 Linköping, Sweden

LINKÖPINGS UNIVERSITET

Datum
Date

2004-01-26

Språk
Language

☐ Svenska/Swedish
☒ Engelska/English
☐ ______________

Rapporttyp
Report category

☐ Licentiatavhandling
☒ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport
☐ ______________

ISBN
—

ISRN
LITH-ISY-EX-3501-2004

Serietitel och serienummer
Title of series, numbering

ISSN
—

URL för elektronisk version

http://www.ep.liu.se/exjobb/isy/2004/3501/

Titel
Title

Estimering av Biasfel med Multipla Plattformar
Multiple Platform Bias Error Estimation

Författare
Author

Åsa Wiklund

Sammanfattning
Abstract

Sensor fusion has long been recognized as a mean to improve target tracking. Sensor fusion deals with the merging of several signals into one to get a better and more reliable result. To get an improved and more reliable result you have to trust the incoming data to be correct and not contain unknown systematic errors. This thesis tries to find and estimate the size of the systematic errors that appear when we have a multi platform environment and data is shared among the units. To be more precise, the error estimated within the scope of this thesis appears when platforms cannot determine their positions correctly and share target tracking data with their own corrupted position as a basis for determining the target's position. The algorithms developed in this thesis use the Kalman filter theory, including the extended Kalman filter and the information filter, to estimate the platform location bias error. Three algorithms are developed with satisfying result. Depending on time constraints and computational demands either one of the algorithms could be preferred.

Nyckelord
Keywords

Sensor fusion, Kalman filter, Bias error, Platform location bias error, Target Tracking

Abstract

Sensor fusion has long been recognized as a mean to improve target tracking. Sensor fusion deals with the merging of several signals into one to get a better and more reliable result. To get an improved and more reliable result you have to trust the incoming data to be correct and not contain unknown systematic errors. This thesis tries to find and estimate the size of the systematic errors that appear when we have a multi platform environment and data is shared among the units. To be more precise, the error estimated within the scope of this thesis appears when platforms cannot determine their positions correctly and share target tracking data with their own corrupted position as a basis for determining the target's position. The algorithms developed in this thesis use the Kalman filter theory, including the extended Kalman filter and the information filter, to estimate the platform location bias error. Three algorithms are developed with satisfying result. Depending on time constraints and computational demands either one of the algorithms could be preferred.

Acknowledgements

This master's thesis completes my Master of Science Degree in Applied Physics and Electrical Engineering at Linköpings universitet. I have many people to thank for my great years at the university. I would not be here without you.

First I would like to thank the Tactical Systems – Decision Support and Autonomous Systems division at SAAB AB in Linköping for giving me the opportunity to do my master's thesis here. Thank you for welcoming me into the group and giving me the chance to see how engineering is performed in real life. I would like to say a special thank you to my two supervisors, M.Sc. Pär Norberg and Ph.D. Måns Mångård, at SAAB AB, you have always supported me with ideas, comments, thoughts and and not least, with help.

Another person whom I want to express my gratitude to is my supervisor at the division of Automatic Control at the Department of Electrical Engineering at Linköpings universitet, M.Sc. Gustaf Hendeby. I have appreciated your profound help with the thesis and its structure and also your ideas within the subject of target tracking.

Finally I would like to thank my family for always giving me the best support in whatever I might be doing. There are a number of persons, not mentioned here, who I also would like to say thank you to, I think you know who you are.

Linköping, January 2004

Åsa Wiklund

Contents

Part I

Introduction

Chapter 1

Thesis Outline

This chapter will introduce the reader to the subject of this thesis. Constraints that have been made to limit the area of investigation and an outline for the thesis will also be presented.

1.1 Background

One of the most modern military aircraft of today is developed and produced by SAAB AB in Linköping. The combat aircraft is named *JAS 39 Gripen* and is used by the Swedish, Hungarian and South African Air Forces. *Gripen* is equipped with state-of-the-art technology and is continuously under development to meet the needs of its customers and to stay competitive as a combat aircraft.

Gripen has fully integrated digital systems which give the aircraft excellent maneuverability and flight performance. *Gripen* has the power to face multiple airborne and/or ground targets or to identify hostile forces during a reconnaissance mission. This is also an area in which new and more advanced technology is being developed.

The state-of-the-art technology of today is to form an *Integrated air picture* which means that several *Gripen* aircrafts are collaborating, together or with other units, to get a better notion of where the enemy is. To make this a reality *Gripen* has the world's most advanced data link which allows the pilot to communicate real-time information with other units and ground control centers while still not revealing their own position to the enemy.

Collaboration to form an *Integrated air picture* involves track and data association of common targets. Measurements are always noisy and it can sometimes be hard to distinguish two targets from each other. With more data from the data link it is possible to estimate the targets' positions better. The data that is communicated via the data link has to be reliable and cannot contain unknown systematic errors. If it does contain systematic errors the *Gripen* association system has to be aware of that and compensate in the association algorithms for the error.

1.2 Task

The purpose of this master's thesis is to estimate the systematic error that is communicated via the data link between platforms.

The objective is also to provide thoughts about how this systematic error might affect the algorithms used for target association.

Algorithms for estimating the systematic error should be developed. The algorithms should then be tested in MATLAB using Monte Carlo simulations.

1.3 Limitations

Since target tracking is a huge area some restrictions have been introduced to limit the problem to a reasonable size. Several assumptions are listed with an explanation of the reason to it.

- The world is two dimensional (flat earth).

  This is to reduce the problem with coordinate conversions.

- The targets are moving but the platforms are stationary.

  Based on the target model, other limitations and that the platforms measure the targets relatively seen from themselves, it will not make a difference if they are moving or not.

- There is only one sensor per platform.

  More sensors give more data, this is just a decision to reduce the amount of data. Sensor fusion could be performed within one platform so more sensors could be used, but to limit the problem this decision was made.

- Perfect association

  The tracking in every platform is considered to be exact. No false sensor tracks appear and no true sensor tracks are missed. The bias error estimation algorithm knows which tracks that belong together. No association has to be done. The issue is discussed in the report.

- The bias error is constant.

  In reality it is drifting very slowly, so it is not a severe restriction.

- All platforms have the same sample rate.

  This is a simplification which allows us to omit the asynchronous case when measures can come at any time or even be absent.

- There is no feedback of data after bias error estimation has taken place.

  This reduces the problem with correlated data. This is more a design feature of the system but could be worth to mention here anyway.

- There is only noise in the target's position, not in the position of the platform.

  The radar is supposed to have much more measurement noise than the navigation system of the platform, except for the bias error. Therefore the latter noise can be neglected without any greater concern.

1.4 Thesis Structure

This master's thesis is written within the context of the Master of Science program in Applied Physics and Electrical Engineering (Y-program) at Linköping University. The thesis gives insight in the subject and provides a solution to a given problem. The aim is that fellow students in the final year of the Y-program should be able to read and understand the content. The chapters in this thesis are organized as follows.

Chapter 1 Thesis Outline (this chapter) introduces the reader to the problem and its limitations. The structure of the thesis is also presented to give a complete picture of what is coming.

Chapter 2 State Estimation provides a theoretical framework for this thesis. The Kalman filter is introduced and so is the extended version of the Kalman filter. Another approach of the Kalman filter theory is presented, the information filter. Finally the least squares estimation is presented.

Chapter 3 Target Tracking gives the reader an insight in target tracking and associated areas. It also brings forward the pros and cons of a multi sensor environment and how to deal with that. The main problem to solve is defined and a discussion of how it affects other related topics is discussed. The coordinate system used to solve the problem is also defined.

Chapter 4 Bias Estimation takes the reader through the different implemented solutions. The chapter also describes the tracking filter and how the different bias estimation algorithms are constructed.

Chapter 5 Simulations discusses the choice of simulation scenarios and parameters. The chapter also describes the design of the system and how the solution is implemented in MATLAB. Plots from MATLAB show the results from the simulations. Finally the result is analyzed.

Chapter 6 Conclusions gives the reader a summary of the thesis and the results obtained. The chapter also discusses conclusions of the work and suggests ideas for further analysis of this topic.

Part II

Theory

Chapter 2

State Estimation

The purpose of target tracking is to estimate certain attributes of the targets and to be able to describe target maneuvers. A common way of doing this is to use a *Kalman filter*. Sometimes the models are non-linear and then an *extended Kalman filter (EKF)* can be used. The theory for the *information filter* is also given as well as the theory for least squares estimation.

2.1 Kalman Filter

The *Kalman filter* requires a state space model to describe the system studied. This gives the advantage of having the signal components as vectors and to be able to handle time-varying systems. The *Kalman filter* was derived by R.E. Kalman in 1960 [10], and has since then been widely used within the area of signal processing. To get more information about the *Kalman filter* read [7] or [8].

The idea behind the Kalman filter is to mix measurements with the estimated value of the signals. The Kalman filter provides an optimal solution to linear Gaussian systems and if the noise is other than Gaussian it will still be the best linear estimator, consult [14]. The filter provides the solution in a minimum variance sense. The Kalman filter state space model is

$$x_{t+1} = A_t x_t + B_t w_t \tag{2.1a}$$
$$y_t = C_t x_t + v_t. \tag{2.1b}$$

The model presented above is discrete but can as well be continuous. How to transform between those two is presented in Section 2.3. In the above equations x_t represents the state and y_t is the measurement vector at time t, both can be vectors. The matrix A_t describes how the states propagate in time and C_t gives the relation between the observed system and the state. The two matrices can be either time dependent or not. The two terms w_t and v_t are white stochastic processes and represents the process and measurement noise. If the Kalman filter equations should hold the following must be true for w_t and v_t

$$\mathsf{E}[w_t] = \mathsf{E}[v_t] = 0 \tag{2.1c}$$
$$\mathsf{E}[w_t w_\tau^T] = \tilde{Q}_t \delta_{t-\tau} \tag{2.1d}$$
$$\mathsf{E}[v_t v_\tau^T] = R_t \delta_{t-\tau} \tag{2.1e}$$
$$\mathsf{E}[w_t v_\tau^T] = 0. \tag{2.1f}$$

The matrix $\tilde{Q}_t$ is time dependent and describes how much we believe in the model (2.1a) used. The matrix R_t is also time dependent and tells how reliable the measurements are. Equation (2.1f) implies that the two noise processes are independent. This is not always the case and then the Kalman filter can still be applied but the equations are not so simple as those presented below.

The Kalman filter equations are:

$$\hat{x}_{t+1|t} = A_t \hat{x}_{t|t} \tag{2.2a}$$
$$P_{t+1|t} = A_t P_{t|t} A_t^T + B_t \tilde{Q}_t B_t^T \tag{2.2b}$$
$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t (y_t - C_t \hat{x}_{t|t-1}) \tag{2.2c}$$
$$P_{t|t} = (I - K_t C_t) P_{t|t-1} \tag{2.2d}$$
$$K_t = P_{t|t-1} C_t^T (C_t P_{t|t-1} C_t^T + R_t)^{-1}. \tag{2.2e}$$

To denote the estimate of the state vector, $\hat{x}_{t|t}$ is used and the true value of the state vector is x_t. The notation $\hat{x}_{t|t}$ means that it is the estimate at time t based on measurements up until time t. If it is $\hat{x}_{t+1|t}$ it means that the estimate $\hat{x}_{t|t}$ is propagated one sampling period. The filter also requires that some initial values are given at time $t = t_0$. These are

$$\hat{x}_{t_0|t_0} = x_0 \tag{2.3a}$$
$$P_{t_0|t_0} = \Pi_0. \tag{2.3b}$$

The initial state estimate is a stochastic variable with expected value x_0. The covariance matrix of this estimate is Π_0 and can be used as a parameter to tune filter performance. More about this in [7, 8] where a proof of the Kalman filter can be found.

2.2 Extended Kalman Filter

The Kalman filter provides an optimal solution to linear Gaussian systems. In practise most models are built from non-linear systems but it would still be an advantage if the Kalman filter equations could be applied. This section will provide means to do this, for more information consult [7, 9, 13, 14].

The method used for this particular application is called *extended Kalman filter*. It can be applied both for non-linearities in the state propagation and in the measurement function. Here only non-linearities in the measurement function will be considered since the state transition function is linear in this application.

The non-linear measurement function can be written as

$$y_t = h_t(x_t) + v_t \tag{2.4}$$

where h_t is a non-linear function describing the relationship between the states and the measurement. With aid from the Jacobian of h_t, in the point $\hat{x}_{t|t-1}$ (the estimate of x_t), the following linear matrix is received and will be denoted

$$H_t := \left.\frac{dh_t}{dx}\right|_{x=\hat{x}_{t|t-1}}. \tag{2.5}$$

The Kalman filter equations are now modified by using H_t in all matrix calculations. In the measurement update $h_t(\hat{x}_{t|t-1})$ can be used instead of $H_t\hat{x}_{t|t-1}$.

The extended Kalman filter equations are

$$\hat{x}_{t+1|t} = A_t\hat{x}_{t|t} \tag{2.6a}$$
$$P_{t+1|t} = A_t P_{t|t} A_t^T + B_t\tilde{Q}_t B_t^T \tag{2.6b}$$
$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t\Big(y_t - h_t(\hat{x}_{t|t-1})\Big) \tag{2.6c}$$
$$P_{t|t} = (I - K_t H_t)P_{t|t-1} \tag{2.6d}$$
$$K_t = P_{t|t-1}H_t^T(H_t P_{t|t-1} H_t^T + R_t)^{-1} \tag{2.6e}$$
$$\hat{x}_{t_0|t_{-1}} = x_0 \tag{2.6f}$$
$$P_{t_0|t_{-1}} = \Pi_0. \tag{2.6g}$$

An alternative to this is to transform the measurements so they can be expressed as a linear combination of the states. After that the ordinary Kalman equations (2.2) can be applied. This transformation requires the same transformation to be applied to R_t. A disadvantage with this approach is that R_t will be more difficult to create since the variances of the measurements will not be as straight forward as before.

2.3 Discretization

To be able to use the discrete version of the extended Kalman filter (2.6), the state space model has to be discrete as well.

This is done with the following transformations. Consider the continous system

$$\dot{x}(t) = Ax(t) + Bw(t) \tag{2.7a}$$
$$y(t) = Cx(t) + v(t) \tag{2.7b}$$

where w is considered to be constant during the sample period, T_s. The the discrete system is given by

$$x_{t+1} = A_t x_t + B_t w_t \tag{2.8a}$$
$$y_t = C_t x_t + v_t \tag{2.8b}$$

where

$$A_t = e^{AT_s}, \qquad B_t = \int_0^{T_s} e^{At} B dt, \qquad C_t = C, \qquad v_t = v(t). \tag{2.9}$$

2.3.1 Continuous Process Noise

In (2.6) $B_t \tilde{Q}_t B_t^T$ is used in the extended Kalman filter equations. Here different approaches are discussed of how to define the state noise when converting from a time continuous to a time discrete situation. In [7, 9] more information about this can be found. Let the following equations describe the situation

$$\dot{x}_t = Ax(t) + Bw(t), \quad \mathsf{Cov}(w(t)) = \tilde{Q}(t) \tag{2.10a}$$
$$x_{t+1} = A_t x_t + B_t w_t, \quad \mathsf{Cov}(w_t) = \tilde{Q}_t. \tag{2.10b}$$

And let $Q_t = B_t \tilde{Q}_t B_t^T$ and $Q(t) = B(t)\tilde{Q}(t)B^T(t)$.

In practise five different alternatives are considered

$$Q_t^a = \int_0^{T_s} e^{A\tau} Q(t) e^{A^T \tau} d\tau \tag{2.11a}$$
$$Q_t^b = \frac{1}{T_s} \int_0^{T_s} e^{A\tau} d\tau Q(t) \int_0^{T_s} e^{A^T \tau} d\tau \tag{2.11b}$$
$$Q_t^c = T_s e^{AT_s} Q(t) e^{A^T T_s} \tag{2.11c}$$
$$Q_t^d = T_s Q(t) \tag{2.11d}$$
$$Q_t^e = T_s A_t Q(t) A_t^T \tag{2.11e}$$

To be able to use the same $Q(t)$ for different sampling intervals all expressions have been normalized with the sampling time T_s. These methods correspond to more or less *ad hoc* assumptions on the state noise for modeling the manoeuvres.

a. w_t is continuous white noise with variance $\tilde{Q}(t)$.

b. $w_t = w_k$ is a stochastic variable which is constant in each sample interval with variance $\tilde{Q}(t)/T_s$. That is, each manoeuvre is distributed over the whole sample interval.

c. w_t is a sequence of Dirac impulses active immediately after a sample is taken. Loosely speaking, we assume $\dot{x} = f(x) + \sum_k w_k \delta_{kT_s - t}$ where w_k is discrete white noise with variance $T_s \tilde{Q}(t)$.

d. w_t is white noise such that its total influence during one sample interval is $T_s\tilde{Q}(t)$.

e. w_t is a discrete white noise sequence with variance $T_s\tilde{Q}(t)$. That is, we assume that all manoeuvres occur suddenly immediately after a sample time, so $x_{t+1} = A_t(x_t + B_t w_t)$.

Note that the first two approaches require a linear time invariant model for the state noise propagation to be exact.

2.4 Information Filter

A disadvantage with the Kalman filter is that if the covariance matrix is large in the beginning, then the Kalman filter can suffer from numerical problems. A solution to this is to propagate the inverse of the covariance matrix instead, to reduce the numerical problems. The technique is referred to as the *Information filter*. Ideas from [11] can provide a further reading. Another reason to choose the *Information filter* is that there is decentralized fusion instead of centralized fusion. References [5] or [7] gives more information regarding this.

The ideas for the information filter is taken from the recursive least squares algorithm. For a derivation of the information filter see [7] or [11]. The information filter equations are presented below. Two auxiliary matrices are introduced to simplify the notation

$$M_t = A_t^{-1} P_{t|t}^{-1} A_t^{-T} \tag{2.12a}$$

$$N_t = A_t B_t (B_t^T A_t B_t + \tilde{Q}_t^{-1})^{-1} \tag{2.12b}$$

using the state space model (2.1). The Kalman filter equations, sorted in time and measurement updates, for the transformed state vector $\hat{a}_{t|k} = P_{t|k}^{-1}\hat{x}_{t|k}$ are

$$\hat{a}_{t+1|t} = (I - N_t B_t^T) A_t^{-1} \hat{a}_{t|t} \tag{2.13a}$$

$$P_{t+1|t}^{-1} = (I - A_t B_t (B_t^T A_t B_t + Q_t^{-1})^{-1}) B_t^T M_t \tag{2.13b}$$

$$= (I - N_t B_t^T) M_t \tag{2.13c}$$

$$\hat{a}_{t|t} = \hat{a}_{t|t-1} + C_t^T R_t^{-1} Y_t \tag{2.13d}$$

$$P_{t|t}^{-1} = P_{t|t-1}^{-1} + C_t^T R_t^{-1} C_t. \tag{2.13e}$$

The following should be noted about the information filter

- $C_t^T R_t^{-1} C_t$ is the information in a new measurement, and $C_t^T R_t^{-1} Y_t$ is sufficient statistic for updating the estimate.

- Vague, or no, prior knowledge of the state can be expressed as $P_0^{-1} = 0$.

2.5 Least Squares Estimation

The least squares estimation assumes a linear relationship between the measurement and the state vector and where the state remains constant over time. The least squares solution is derived by [3] and a useful extension of that is:

$$R = \left[\sum_i R_i^{-1} \right]^{-1}, \quad \hat{\Delta} = R \sum_i R_i^{-1} \hat{\Delta}_i \tag{2.14}$$

where Δ is the raw bias measurement and R is the measurement covariance matrix. The estimation errors can not be correlated if the above relationships should hold. This assumption typically holds when using newly acquired measurement data.

Chapter 3

Target Tracking

Target tracking means that for example an aircraft gathers information about other objects (targets) in the surroundings and tries to make an accurate picture of the scenario. Things that can be of interest is where the targets are located, how they are moving and/or what type of objects there are.

The information is gathered by different types of sensors, for example, radar, IR, active and passive sonar and laser radar.

The collected information can for example be the distance and the bearing to a target, which will be the case in this report. With the aid of these measurements the target's state is estimated. A target's state involves, usually, position and velocity. If the uncertainty is included in the estimate it is called a target's track instead.

This chapter will introduce a reference frame for the problem to be solved and discuss target tracking with multiple sensors. The association problem in target tracking will also be mentioned before sensor bias is discussed both in a general and in the specific case.

3.1 Coordinate System

In target tracking and state estimation of targets different coordinate systems can be used, see for example [15]. In this application a two dimensional flat-earth coordinate system is used. The coordinates can be centered at the different platforms or be global with a common origin for all the platforms. The platforms can determine target position in either polar coordinates or Cartesian coordinates. In the global coordinate system positions are defined in Cartesian coordinates.

3.2 Target Tracking With Multiple Sensors

When using multiple sensors in target tracking it is of great interest to improve estimation of targets by combining data from all sensors. The topic is often referred to as the *sensor fusion* problem. The sensor fusion problem have two different approaches, either there is sensor redundancy so the problem could be solved by using any of the sensors, or each sensor provides unique information.

The question arises concerning the best way of using multiple sensors to improve target tracking. The following [1, 7, 12] are discussing this issue.

One way is to let all the platforms send raw data to a central processing unit. In the central processing unit all the measurements are used to form data association and estimation of states. This is called a centralized structure. On the other hand, using a decentralized structure, each sensor carries out estimation locally by use of a filter and then submits the tracks to the central processing unit.

So in central fusion there is access to all measurements when deriving the optimal filter. In decentralized fusion, on the other hand, a filter is applied to each measurement and the fusion process only have access to the state estimates and the covariance matrices, see Figure 3.1.

Both approaches requires good communication between the sensors and the central processing unit, because a lot of data is being transfered. The centralized structure demands the central processing unit to handle a lot of data and perform many computations. In the decentralized structure the state estimate and the covariance matrix is being sent to the central processor. This usually means more data than in the measurement vector but on the other hand less signaling may be needed.

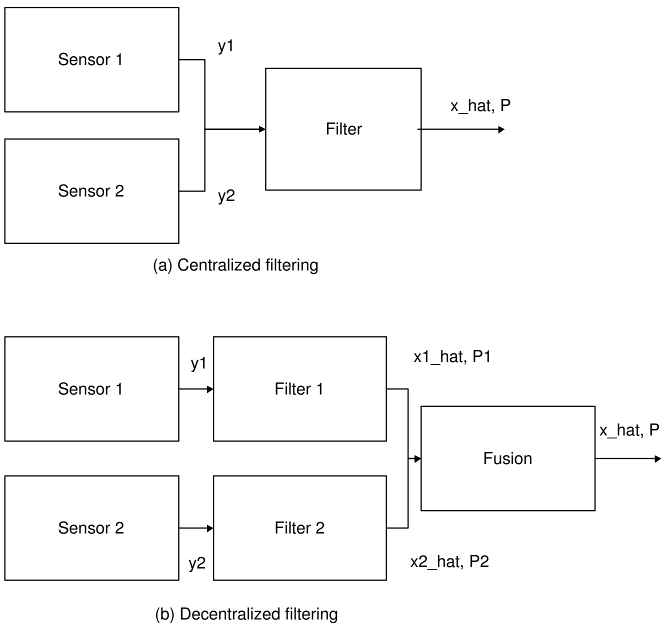


Figure 3.1. Centralized and decentralized filtering. y are the measurements and $\hat{x}$ are the state estimates with covariance matrices P.

3.2.1 Centralized Fusion

In a centralized structure one can use the Kalman filter theory to collect all measurements in one measurement equation. One of the great advantages with the Kalman filter is to be able to handle vector valued signals. The model can be as follows

$$y_t = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_m \end{pmatrix} x_t + \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{pmatrix}. \tag{3.1}$$

The problem that can arise is if different sensors work with different coordinates. For example, if one sensor submits the information in Cartesian coordinates and another sensor delivers the information in polar coordinates. An extended Kalman filter can handle this type of problem, see Section 2.2.

3.2.2 Decentralized Fusion

If the structure of the system is decentralized the central processor receives two independent state estimates $\hat{x}^1$ and $\hat{x}^2$, with covariance matrices P^1 and P^2. According to [7] fusion of these are straight-forward

$$\hat{x} = P\Big((P^1)^{-1}\hat{x}^1 + (P^2)^{-1}\hat{x}^2\Big) \tag{3.2a}$$

$$P = \Big((P^1)^{-1} + (P^2)^{-1}\Big)^{-1}. \tag{3.2b}$$

Suppose there is a situation when two Kalman filters are working with the same state vector. The total state space model for the decentralized filters can be written as

$$\begin{pmatrix} x^1_{t+1} \\ x^2_{t+1} \end{pmatrix} = \overbrace{\begin{pmatrix} A_t & 0 \\ 0 & A_t \end{pmatrix}}^{\bar{A}} \begin{pmatrix} x^1_t \\ x^2_t \end{pmatrix} + \overbrace{\begin{pmatrix} B_t & 0 \\ 0 & B_t \end{pmatrix}}^{\bar{B}} \begin{pmatrix} w^1_t \\ w^2_t \end{pmatrix} \tag{3.3a}$$

$$y_t = \overbrace{\begin{pmatrix} C^1_t & 0 \\ 0 & C^2_t \end{pmatrix}}^{\bar{C}} \begin{pmatrix} x^1_t \\ x^2_t \end{pmatrix} + \begin{pmatrix} v^1_t \\ v^2_t \end{pmatrix} \tag{3.3b}$$

$$\mathsf{Cov}(w^i_t) = \tilde{Q}_t \tag{3.3c}$$

Since the $\bar{A}$, $\bar{B}$ and $\bar{C}$ matrices are block diagonal

- The updates of the estimates can be done separately, and hence decentralized.
- The state estimates will be independent.

Since there is independence between the state estimates the fusion formula (3.2) can be applied. According to [7] the combined state space model for the decentralized filter and the fusion filter becomes

$$\begin{pmatrix} x_{t+1}^1 \\ x_{t+1}^2 \\ x_{t+1} \end{pmatrix} = \begin{pmatrix} A_t & 0 & 0 \\ 0 & A_t & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_t^1 \\ x_t^2 \\ x_t \end{pmatrix} + \begin{pmatrix} B_t & 0 & 0 \\ 0 & B_t & 0 \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} w_t^1 \\ w_t^2 \\ w_t^3 \end{pmatrix} \tag{3.4a}$$

$$\begin{pmatrix} y_t^1 \\ y_t^2 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} C_t^1 & 0 & 0 \\ 0 & C_t^2 & 0 \\ I & 0 & -I \\ 0 & I & -I \end{pmatrix} \begin{pmatrix} x_t^1 \\ x_t^2 \\ x_t \end{pmatrix} + \begin{pmatrix} v_t^1 \\ v_t^2 \\ 0 \\ 0 \end{pmatrix} \tag{3.4b}$$

$$\mathsf{Cov}(w_t^1) = \tilde{Q}_t \tag{3.4c}$$

$$\mathsf{Cov}(w_t^2) = \tilde{Q}_t \tag{3.4d}$$

$$\mathsf{Cov}(w_t^3) = \infty \cdot I \tag{3.4e}$$

If the Kalman filter is applied to this state space model then the fusioned state estimate will be found as the third element of the state vector.

The problem with this formulation is that we have *lost information*. The decentralized filters do not use the fact that they are estimating the same process and thereby have the same state noise. The covariance matrix will become too small, compared to that which the Kalman filter gives for the model

$$x_{t+1} = A_t x_t + B_t w_t \tag{3.5a}$$

$$y_t = \begin{pmatrix} C_t^1 \\ C_t^2 \end{pmatrix} x_t + \begin{pmatrix} v_t^1 \\ v_t^2 \end{pmatrix} \tag{3.5b}$$

$$\mathsf{Cov}(w_t) = \tilde{Q}_t. \tag{3.5c}$$

If the aim is to get an optimal state estimate under the given premises the Kalman filter has to be inverted to recover the raw information in the measurements. One solution to the problem could be to try to estimate, or to ask for, the Kalman filter transfer function in all decentralized filters. In some cases it could be enough to estimate the Kalman gain in the decentralized filters. Another alternative, which in most cases also is a better alternative, is to use the theory of the *Information filter* described in Section 2.4. The measurement update equations of the information filter was

$$\hat{a}_{t|t} = \hat{a}_{t|t-1} + C_t^T R_t^{-1} Y_t \tag{3.6a}$$

$$P_{t|t}^{-1} = P_{t|t-1}^{-1} + C_t^T R_t^{-1} C_t. \tag{3.6b}$$

By rewriting the equations the information in each measurement is recovered and the formulas for decentralized filtering is given by

$$C_t^T R_t^{-1} Y_t = \hat{a}_{t|t} - \hat{a}_{t|t-1} = (\hat{P}_{t|t})^{-1} \hat{x}_{t|t} - (\hat{P}_{t|t-1})^{-1} \hat{x}_{t|t-1} \tag{3.7a}$$

$$C_t^T R_t^{-1} C_t = \hat{P}_{t|t}^{-1} - \hat{P}_{t|t-1}^{-1}. \tag{3.7b}$$

And when many tracks are fused together the optimal sensor fusion is given by

$$\hat{P}_{t|t}^{-1}\hat{x}_{t|t} = \hat{P}_{t|t-1}^{-1}\hat{x}_{t|t-1} + \sum_{i=1}^{m}\Big((\hat{P}_{t|t}^{i})^{-1}\hat{x}_{t|t}^{i} - (\hat{P}_{t|t-1}^{i})^{-1}\hat{x}_{t|t-1}^{i}\Big) \tag{3.8a}$$

$$\hat{P}_{t|t}^{-1} = \hat{P}_{t|t-1}^{-1} + \sum_{i=1}^{m}\Big((\hat{P}_{t|t}^{i})^{-1} - (\hat{P}_{t|t-1}^{i})^{-1}\Big). \tag{3.8b}$$

3.3 Track Association

Track association, or data association, appears when several platforms are estimating several targets and when the platforms share data in some way. Track association can occur in different forms. Here the general case, with no bias error present, is considered.

- association of measurements to tracks
- association of measurements to measurements, to initiate tracks
- association of tracks to tracks, which will be the case here

The central processor receives sensor tracks and tries to decide which sensor tracks that belong to a specific central track. A sensor track is the data collection that a single sensor produces and a central track is the joint track from a number of sensor tracks.

One approach to solve this association problem is to use the nearest neighbor method. The idea of the method is to merge the sensor track that is closest to the central track with the central track. A condition is that there has to be a one-to-one relationship between sensor tracks and central tracks. The best possible solution is to minimize some kind of defined distance. Figure 3.2 shows the situation.

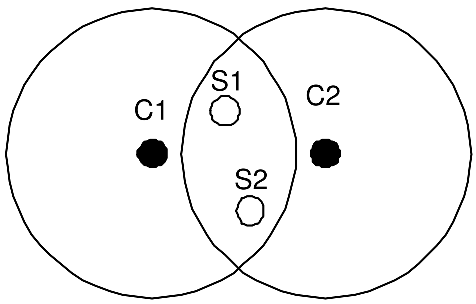


Figure 3.2. Two sensor tracks (S1 and S2) and two central tracks (C1 and C2) The two circles show the gates belonging to the two central tracks.

To remove many irrelevant sensor tracks gating is used before the association algorithm is applied. A gate is a limit set to discard the tracks that are outside and include the tracks that are inside the gate. The gate is estimated in a clever way so that possible sensor tracks are not discarded. To gain a deeper knowledge in the association problem, see [12].

3.3.1 Sensor Bias and Association

In the track association presented in Section 3.3 no systematic error is present in the sensor tracks. If there is a systematic error, a sensor bias error, it has to be taken in account when performing track association. For example, a more allowing or larger gate may need to be used since the sensor bias might have repositioned the sensor track. The nearest neighboour algorithm have to be adjusted to fit the new conditions. The algorithm has to be combined with some bias estimation process and the tracks have to be adjusted for the estimated sensor bias.

In this case there is no central track that the sensor tracks should be fused with. Instead different sensor tracks are associated with each other and a bias error is estimated in between them. So it might not be the two closest tracks that are associated with each other. Instead two other tracks might belong to each other and be closest when a bias error is estimated and compensated for.

Figure 3.3(a) shows the case if no bias error is estimated and the two closest tracks are associated with each other. Figure 3.3(b) describes the situation when a bias error is estimated and compensated for before association is performed. The result here is different. When there is a bias error present some kind of hypothesis test might be needed to find which is the most probable solution to the tracking situation. Either there is a bias error present or not, and if there is a bias error present, what is the size of it.

(a) (b)

Figure 3.3. Sensor tracks are associated with each other. Figure 3.3(a) shows the case if no bias error is estimated and the two closest tracks are associated with each other. Figure 3.3(b) describes the situation when a bias error is estimated and compensated for before association is performed. Different solutions appear depending on if there is a bias error present or not.

It should be pointed out that by performing bias estimation and association in the way described above can lead to disastrous results as well. If a bias error is estimated and compensated for and then the sensor tracks are fused together there may be even stronger belief in that the correct bias error is estimated. But the estimated bias error might be wrong and still this method believes very strong in it. Somehow a joint bias-association algorithm has to be constructed which puts a relevant amount of trust in the estimated bias error.

3.4 Sensor Bias

As mentioned in the previous section, *sensor bias*, or sensor registration errors as it also can be called, are important issues to consider when performing track association and sensor data fusion. The definition of a sensor bias is that there is a systematic error that appears in comparison with the random errors that are almost always present. This problem is discussed in a number of books and articles, some of them are [1, 2, 3, 4, 6, 15].

There are many different kinds of sensor biases that depends on the specific situation or application. For example, the targets and/or platforms can be moving or be stationary. There can be several sensors on one platform. The sensors can be of the same type or of different types. All these different combinations lead to different kinds of biases.

Sensor biases that appear in these situations are

- Measurement biases
- Time biases
- Measurement axes misalignment biases
- Sensor (or platform) location biases
- Refraction biases
- Attitude biases
- Coordinate conversion biases

3.4.1 Platform Location Bias

This thesis aims to resolve the problem with the *platform location bias*.

Several platforms are tracking several targets and in order to gain better estimates of the targets' states the platforms share data. A platform cannot determine its position with high enough precision so there will be a *platform location bias* present in the estimate. When the platforms share data they have to use a common reference frame where the positions are to be determined, see Section 3.1. When the platforms transform target position in their reference frame to the global reference frame the platform location bias will be incorporated within the position of the target.

When another platform receives the data it will be transformed to that specific platforms reference frame. The platform receiving the data will now have two estimates of the target's position. The estimates will differ because of the systematic bias error and other random errors.

The bias error that is possible to estimate is the relative error that appears between the two position estimates. It is not possible to observe the absolute bias error belonging to either one of the platforms, unless one of them has zero bias. See Figure 3.4 for a description of the situation.

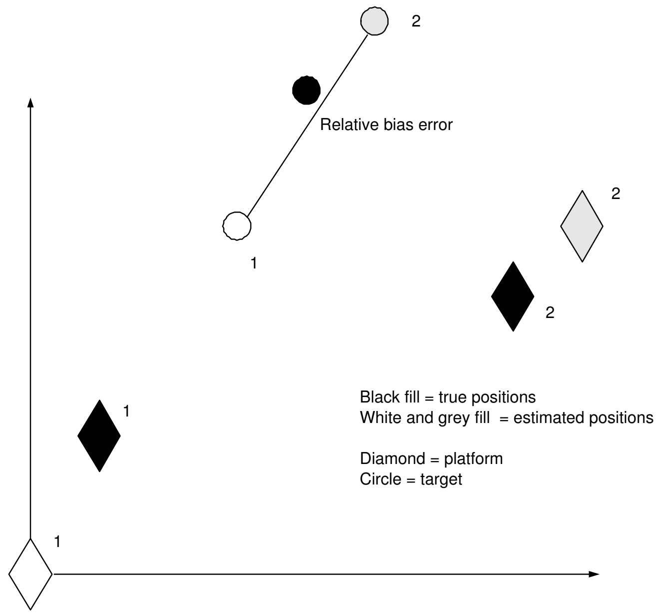


Figure 3.4. The relative bias error that can be estimated by the platform that receives tracking data from another platform.

Chapter 4

Bias Estimation

This chapter describes the different approaches to platform location bias error estimation. The solution is implemented in MATLAB, which is a well known and easy to use tool. To be as realistic as possible there are tracking filters before the bias estimation process takes place. The chapter begins with a short description of the whole tracking process.

4.1 Tracking Process

Each platform has a tracking filter that holds a track for every target it measures. The platforms communicate their tracking data to other platforms so one platform receives tracking data from all the others. Figure 4.1 gives a schematic view of the situation.

When a track is passed on from a platform it is converted from the local coordinates in the platform to the global coordinates described in Section 3.1. When conversion between the coordinate systems is done the platform uses its own (biased) position. The global estimate of the track will include the bias error. This bias error can be estimated in several different ways which will be described in the following sections. The best estimate that could possibly be done, if there is access to the raw measurements, is to do a least squares estimate for the bias error. See Section 2.5 for the prerequisites needed. In this application it is assumed that the raw measurements cannot be reached and filtered data has to be used. Therefore the least squares estimate cannot be used and other solutions have to be found.

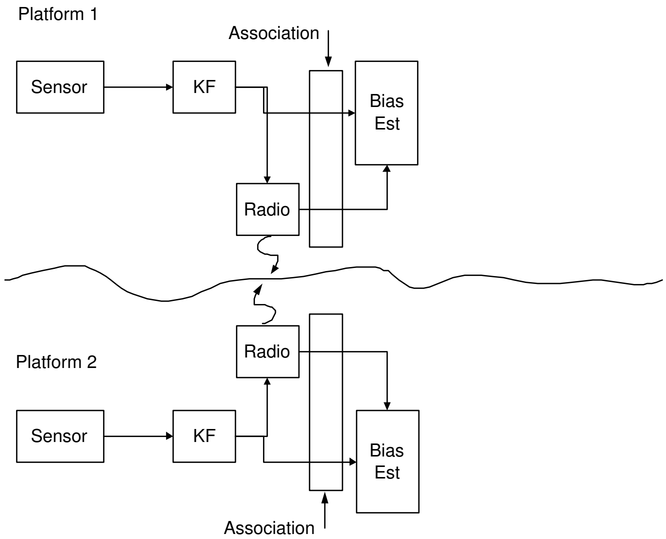


Figure 4.1. This figure shows the situation when two platforms are communicating their target tracks to each other.

4.2 Tracking Filter

A possible state space model for the tracking filters, that will be used here, is as follows. The state vector is defined as

$$x_t = \begin{pmatrix} \mathsf{x} \\ \mathsf{y} \\ \mathsf{v}_x \\ \mathsf{v}_y \end{pmatrix}. \tag{4.1}$$

And the state space model is

$$x_{t+1} = \begin{pmatrix} 1 & 0 & T_s & 0 \\ 0 & 1 & 0 & T_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} x_t + B_t w_t \tag{4.2a}$$

$$y_t = \begin{pmatrix} \sqrt{\mathsf{x}^2 + \mathsf{y}^2} \\ \arctan(\frac{\mathsf{y}}{\mathsf{x}}) \end{pmatrix} + \begin{pmatrix} v_r \\ v_\varphi \end{pmatrix}. \tag{4.2b}$$

In order to apply the non-linear Kalman filter stated in equation (2.6) the measurement equation has to be linearized. Also recall the situation in (2.4). The

Jacobian is

$$H_t = \frac{dh_t}{dx_t}\bigg|_{x=\hat{x}_{t|t-1}} = \mathcal{J}\begin{pmatrix} \sqrt{\mathsf{x}^2+\mathsf{y}^2} \\ \arctan(\frac{\mathsf{y}}{\mathsf{x}}) \end{pmatrix}\Bigg|_{x=\hat{x}_{t|t-1}} =$$

$$= \begin{pmatrix} \frac{\mathsf{x}}{r} & \frac{\mathsf{y}}{r} & 0 & 0 \\ \frac{-\mathsf{y}}{r^2} & \frac{\mathsf{x}}{r^2} & 0 & 0 \end{pmatrix}\Bigg|_{x=\hat{x}_{t|t-1}}$$

where

$$r = \sqrt{\mathsf{x}^2+\mathsf{y}^2}.$$

Based on the discussion in Section 2.3, the Q_t and R_t matrices are as follows

$$Q_t = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{4.3}$$

$$R_t = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\varphi^2 \end{pmatrix} \tag{4.4}$$

where $\sigma_r^2 = E[v_r v_r^T]$ and $\sigma_\varphi^2 = E[v_\varphi v_\varphi^T]$. The initial conditions Π_0 and x_0 are set to be

$$\Pi_0 = \begin{pmatrix} 100^2 & 0 & 0 & 0 \\ 0 & 100^2 & 0 & 0 \\ 0 & 0 & 10^2 & 0 \\ 0 & 0 & 0 & 10^2 \end{pmatrix} \tag{4.5}$$

$$x_0 = \begin{pmatrix} \mathsf{x}_0 \\ \mathsf{y}_0 \\ \mathsf{v}_{0x} \\ \mathsf{v}_{0y} \end{pmatrix} \tag{4.6}$$

where x_0 and y_0 define the initial positions of the measured target track and v_{0x} v_{0y} are the initial velocities for the target.

4.3 Basic Kalman Filter

The first approach to bias estimation is to have a Kalman filter that estimates the bias between the two different platforms.

The state vector is defined to be

$$\Delta_t^{ij} = \begin{pmatrix} \mathsf{x}^i - \mathsf{x}^j \\ \mathsf{y}^i - \mathsf{y}^j \end{pmatrix} \tag{4.7}$$

where the exponents indicate platform i and platform j. It could be any pair of two platforms.

The state space model for the bias estimation is

$$\Delta_{t+1}^{ij} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \Delta_t^{ij} \tag{4.8a}$$

$$z_t = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \Delta_t^{ij} + e_{\Delta t} \tag{4.8b}$$

The matrices that describes the process and measurement noise are here defined to be

$$Q_\Delta = \begin{pmatrix} 10 & 0 \\ 0 & 10 \end{pmatrix} \tag{4.9}$$

$$R_\Delta = P^i + P^j \tag{4.10}$$

where P^i and P^j are the covariance matrices from the two track filters in platform i and j. The reason to choose $P^i + P^j$ as R_Δ is that P^i and P^j reflect the uncertainty in the incoming state estimates. According to (4.8a) there is no process noise but simulations show that a Q_Δ gives better performance of the estimation algorithm. The bias reaches a steady level faster than without a Q_Δ matrix.

According to the measurement equation (4.8b) there is measurement noise present. Actually there is no explicit measurement noise, but instead the measurement noise is embedded in the state estimates from the tracking filters. By just adding the two covariance matrices an error is made, see Appendix B.

The initial conditions in the bias filter are

$$P_0^\Delta = P^i + P^j \tag{4.11}$$

and

$$\Delta_0^{ij} = \text{first measurement.} \tag{4.12}$$

The same error, as in (4.10), is made in (4.11). P^i and P^j are chosen here of the same reason as before.

A pair of platforms, in this case platform i and j, can track several common targets. If this is the case it is possible to update the bias estimate with several measurements. So after one time update the Kalman filter iterates the measurement equation over the number of bias measurements that are available. Since the different platforms are estimating the same target one could guess that the different measurements will be correlated. If there is correlation involved it is neglected here. Later on, see Section 4.5, disadvantages with this method will be presented and a motivation for that this type of correlation which is neglected here will diminish in regard to other errors made.

4.4 Blackman Model

Another approach that is very similar to the one presented in Section 4.3 uses ideas from [3]. The state space model for the bias filter is the same but instead of iterating over the measurement equation in the Kalman filter over and over again the relationship (4.13) is used to estimate the bias. The relation is normally used to combine two sensor-level track state estimates to get a combined state estimate that minimizes the expected error. The approach here is that it would work well in combining bias state estimates.

There are two platforms estimating several common targets. The platforms are, as before, numbered i, j and so on. The targets are named with capital letters A, B etc. There is now a bias filter for every pair of platforms for every target. The same bias filter as in the Kalman filter method is used here. The bias is fused according to

$$\hat{\Delta}^{ij} = \hat{\Delta}^{ij}_A + \Psi[\hat{\Delta}^{ij}_B - \hat{\Delta}^{ij}_A] \tag{4.13}$$

where

$$\Psi = [P^{\Delta}_A - P^{\Delta}_{AB}]U^{-1}_{AB} \tag{4.14}$$

and

$$U_{AB} = P^{\Delta}_A + P^{\Delta}_B - P^{\Delta}_{AB} - {P^{\Delta}_{AB}}^T. \tag{4.15}$$

P^{Δ}_A and P^{Δ}_B are the covariance matrices from the two bias filters. P^{Δ}_{AB} is the cross-covariance matrix that defines the error correlation between the two bias estimates. The cross-covariance is very hard to determine and is therefore set to zero in this application. The cross-covariance could be determined in a recursive manner with the initial condition set to zero. But if the process noise is zero, as in this application, the cross-covariance will remain zero.

The covariance matrix for the fused bias estimate is

$$P^{\Delta}_f = P^{\Delta}_A - [P^{\Delta}_A - P^{\Delta}_{AB}]U^{-1}_{AB}[P^{\Delta}_A - P^{\Delta}_{AB}]^T. \tag{4.16}$$

In (4.13) the bias from targets A and B are fused. When receiving measurements from a target C the bias estimate becomes

$$\hat{\Delta}^{ij} = \hat{\Delta}^{ij}_{old} + \Psi[\hat{\Delta}^{ij}_C - \hat{\Delta}^{ij}_{old}] \tag{4.17}$$

with

$$\Psi = [P^{\Delta}_f - P^{\Delta}_{fC}]U^{-1}_{fC} \tag{4.18}$$

and

$$U_{fC} = P^{\Delta}_f + P^{\Delta}_C - P^{\Delta}_{fC} - {P^{\Delta}_{fC}}^T \tag{4.19}$$

where $\hat{\Delta}^{ij}_{old}$ means the previous estimated bias error.

The covariance matrix is

$$P^{\Delta}_f = P^{\Delta}_{f_{old}} - [P^{\Delta}_{f_{old}} - P^{\Delta}_{fC}]U^{-1}_{fC}[P^{\Delta}_{f_{old}} - P^{\Delta}_{fC}]^T \tag{4.20}$$

where $P^{\Delta}_{f_{old}}$ is the previous estimated covariance matrix.

Considering this application with the cross-covariances set to zero we have the following set of equations

$$\hat{\Delta}^{ij} = \hat{\Delta}^{ij}_A + \Psi[\hat{\Delta}^{ij}_B - \hat{\Delta}^{ij}_A] \tag{4.21}$$

where

$$\Psi = [P^{\Delta}_A]U^{-1}_{AB} \tag{4.22}$$

and

$$U_{AB} = P_A^{\Delta} + P_B^{\Delta}. \tag{4.23}$$

The fused covariance becomes

$$P_f^{\Delta} = P_A^{\Delta} - [P_A^{\Delta}]U_{AB}^{-1}[P_A^{\Delta}]^T. \tag{4.24}$$

When updating the bias estimate with C we have

$$\hat{\Delta}^{ij} = \hat{\Delta}_{old}^{ij} + \Psi[\hat{\Delta}_C^{ij} - \hat{\Delta}_{old}^{ij}] \tag{4.25}$$

with

$$\Psi = [P_f^{\Delta}]U_{fC}^{-1} \tag{4.26}$$

and

$$U_{fC} = P_f^{\Delta} + P_C^{\Delta}. \tag{4.27}$$

The covariance matrix is

$$P_f^{\Delta} = P_{f_{old}}^{\Delta} - [P_{f_{old}}^{\Delta}]U_{fC}^{-1}[P_{f_{old}}^{\Delta}]^T. \tag{4.28}$$

4.5 Ad Hoc Solution

The methods described in Section 4.3 and Section 4.4 have certain disadvantages. Since the measurements to the bias filter are already filtered there could be a lot of correlation between $\hat{x}_t^i$ and $\hat{x}_{t+1}^i$. To solve this problem one would like to know if the state $\hat{x}_t^i$ has been updated from new measurement or from just model propagation in time. If it has been updated by new measurements the two consecutive states are probably not so correlated and can be used as they are. But if only, or a lot of, model propagation has taken place and there is very little measurement update the uncertainty of the bias estimate should stay the same and the bias state vector should not be updated.

The problem could then be solved by comparing the states $A_t\hat{x}_t^i$ and $\hat{x}_{t+1}^i$ to see if there is any difference between what the model predicts and what the new state is. Some parameter that tells if the new state can be trusted or not, could maybe be estimated in a clever way. The bias filter can then include this parameter which will control the update of the bias estimate. This solution has not been implemented since another approach, see Section 4.6, turned up and solved the same problem. As discussed in 4.1 the best possible solution would be to use the raw measurements directly. The *Information filter* provides a solution where information from the raw measurements is used.

4.6 Information Filter

The *Information filter* described in Section 2.4 provides another solution to the correlation problem presented in Section 4.5.

The information filter takes into account that the two platforms are estimating the state of the same target so there is only one process noise involved.

The idea is to propose a disired model and then rewrite the equations on a form so the information filter can be used. The following model is what we want to have.

The state vector is

$$\Phi = \begin{pmatrix} x_t \\ \Delta^{ij} \end{pmatrix} \tag{4.29}$$

where x is as in (4.1) and Δ^{ij} as in (4.7). The state space model is defined as

$$\Phi_{t+1} = \begin{pmatrix} x_{t+1} \\ \Delta^{ij}_{t+1} \end{pmatrix} = \begin{pmatrix} A_t & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} x_t \\ \Delta^{ij}_t \end{pmatrix} + w_t \tag{4.30a}$$

$$Y_t = \begin{pmatrix} y^i_t \\ y^j_t \end{pmatrix} = \begin{pmatrix} C^i_t & 0 \\ C^j_t & -I \end{pmatrix} \begin{pmatrix} x_t \\ \Delta^{ij}_t \end{pmatrix} + \begin{pmatrix} v^i_t \\ v^j_t \end{pmatrix} \tag{4.30b}$$

A_t is the matrix in (4.2a), $I_{2\times 2}$ is the identity matrix and the C_t matrices are defined as

$$C^i_t = C^j_t = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{4.31}$$

The measurement and process noise matrices are

$$Q_t = \begin{pmatrix} Q_t & 0 \\ 0 & Q^{\Delta}_t \end{pmatrix} \tag{4.32}$$

$$R_t = \begin{pmatrix} R^i_t & 0 \\ 0 & R^j_t \end{pmatrix} \tag{4.33}$$

The measurements are in reality in r and φ coordinates but let us believe that this is unknown. Think of it as having a black box with measurements as output. Instead the model is built as if the measurements would have been in x and y coordinates. The reason for doing this is that the measurements are not received as input but instead the state estimates are received, from the tracking filters in each platform.

The information filter measurement update equations are

$$\hat{a}_{t|t} = \hat{a}_{t|t-1} + C^T_t R^{-1}_t Y_t \tag{4.34a}$$

$$\hat{P}^{-1}_{t|t} = \hat{P}^{-1}_{t|t-1} + C^T_t R^{-1}_t C_t \tag{4.34b}$$

where $\hat{a}_{t|t} = \hat{P}^{-1}_{t|t} \hat{x}_{t|t}$.

By using the structure of the matrices involved it can be shown that, even if the information filter requires the same filter to be used both in the distributed filters and in the central filter, the bias can be left out in the distributed filters. This is quite intuitive since one can not know the bias by just using the measurements from one platform.

$$C^T_t R^{-1}_t = \begin{pmatrix} C^i_t & 0 \\ C^j_t & -I \end{pmatrix}^T \begin{pmatrix} R^i_t & 0 \\ 0 & R^j_t \end{pmatrix}^{-1} = \begin{pmatrix} (C^i_t)^T (R^i_t)^{-1} & (C^j_t)^T (R^j_t)^{-1} \\ 0 & -(R^j_t)^{-1} \end{pmatrix} \tag{4.35}$$

From this it follows that

$$\hat{a}_{t|t} = \hat{a}_{t|t-1} + C_t^T R_t^{-1} Y_t = \hat{a}_{t|t-1} + \begin{pmatrix} (C_t^i)^T (R_t^i)^{-1} y_t^i + (C_t^j)^T (R_t^j)^{-1} y_t^j \\ -(R_t^j)^{-1} y_t^j \end{pmatrix} \tag{4.36}$$

and

$$\begin{aligned} \hat{P}_{t|t}^{-1} &= \hat{P}_{t|t-1}^{-1} + C_t^T R_t^{-1} C_t = \\ &= \hat{P}_{t|t-1}^{-1} + \begin{pmatrix} (C_t^i)^T (R_t^i)^{-1} C_t^i + (C_t^j)^T (R_t^j)^{-1} C_t^j & -(C_t^j)^T (R_t^j)^{-1} \\ -(R_t^j)^{-1} (C_t^j) & (R_t^j)^{-1} \end{pmatrix} \end{aligned} \tag{4.37}$$

We can find two of the measurement update quantities according to

$$(C_t^i)^T (R_t^i)^{-1} y_t^i = (P_{t|t}^i)^{-1} \hat{x}_{t|t}^i - (P_{t|t-1}^i)^{-1} \hat{x}_{t|t-1}^i \tag{4.38a}$$

$$(C_t^i)^T (R_t^i)^{-1} C_t^i = (P_{t|t}^i)^{-1} - (P_{t|t-1}^i)^{-1} \tag{4.38b}$$

The remaining problem is to find the quantities

$$-(R_t^j)^{-1} y_t^j \tag{4.39a}$$

$$-(C_t^j)^T (R_t^j)^{-1} \tag{4.39b}$$

$$-(R_t^j)^{-1} (C_t^j) \tag{4.39c}$$

$$(R_t^j)^{-1} \tag{4.39d}$$

Consider $(C_t^j)^T (R_t^j)^{-1} y_t^j$ and expand it

$$(C_t^j)^T (R_t^j)^{-1} y_t^j = \begin{pmatrix} I \\ 0 \end{pmatrix} (R_t^j)^{-1} y_t^j = \begin{pmatrix} (R_t^j)^{-1} y_t^j \\ 0 \end{pmatrix} \tag{4.40}$$

and the second block row in (4.36) is found.

The three elements $(1,2)$, $(2,1)$ and $(2,2)$ in (4.37) can be derived by considering

$$(C_t^j)^T (R_t^j)^{-1} C_t^j = (C_t^j)^T (R_t^j)^{-1} \begin{pmatrix} I & 0 \end{pmatrix} = \begin{pmatrix} (C_t^j)^T (R_t^j)^{-1} & 0 \end{pmatrix} \tag{4.41}$$

which gives us element $(1,2)$ (except for the negative sign),

$$(C_t^j)^T (R_t^j)^{-1} C_t^j = \begin{pmatrix} I \\ 0 \end{pmatrix} (R_t^j)^{-1} C_t^j = \begin{pmatrix} (R_t^j)^{-1} C_t^j \\ 0 \end{pmatrix} \tag{4.42}$$

which gives us element $(2,1)$ (except for the negative sign)

$$(C_t^j)^T (R_t^j)^{-1} C_t^j = \begin{pmatrix} I \\ 0 \end{pmatrix} (R_t^j)^{-1} \begin{pmatrix} I & 0 \end{pmatrix} = \begin{pmatrix} (R_t^j)^{-1} & 0 \\ 0 & 0 \end{pmatrix} \tag{4.43}$$

and finally element $(2,2)$ is derived.

4.6.1 Non-linear Transformation

By using the special structure of the matrices above the quantities needed can be found without the use of the nonlinear transformation of the matrix R_t.

As said before the measurements are believed to be in x and y coordinates but in fact they are in r and φ coordinates. The measurement noise, expressed in R_t is given in the polar coordinates and has to be converted to Cartesian coordinates. This is done by a non-linear transformation which is just performed approximately, see [3].

By using the coordinate conversion

$$\mathsf{x} = r\cos\varphi, \quad \mathsf{y} = r\sin\varphi \tag{4.44}$$

we receive the following matrix

$$R_t = \begin{pmatrix} \sigma_{\mathsf{x}}^2 & \sigma_{\mathsf{xy}}^2 \\ \sigma_{\mathsf{xy}}^2 & \sigma_{\mathsf{y}}^2 \end{pmatrix} \tag{4.45}$$

where

$$\begin{aligned} \sigma_{\mathsf{x}}^2 &= \sigma_r^2 \cos^2\varphi + \sigma_\varphi^2 r^2 \sin_\varphi^2 \\ \sigma_{\mathsf{y}}^2 &= \sigma_r^2 \sin^2\varphi + \sigma_\varphi^2 r^2 \cos_\varphi^2 \\ \sigma_{\mathsf{xy}}^2 &= \frac{1}{2}\sin(2\varphi)[\sigma_r^2 - r^2\sigma_\varphi^2]. \end{aligned}$$

This approximation is considered to be accurate when $r\sigma_\varphi^2/\sigma_r < 0.4$ and when $\sigma_\varphi < 0.4$ rad which is the normal case.

Part III

Results

Chapter 5

Simulations

The chapter begins with a description of how the test environment is constructed. The scenarios used for the simulations are presented and the performance of the tracking filter is shown. Results from the bias estimation process are presented and discussed. The solution is implemented in MATLAB, which is a well known and easy to use tool.

5.1 Design and Data Generation

In order to simulate the bias estimation process as close to reality as possible the following test environment has been built. First data generation is performed where the target tracks are generated from a straight-line model with constant velocity. The data that has been generated is distance and azimuth angle seen from each platform to each target. Following this are the tracking filters which send their processed data to a common reference frame. The bias error estimation process is performed and finally there is an evaluation of the algorithm. All parts have been implemented in MATLAB because MATLAB provides a simple interface but still have powerful tools. One disadvantage might be that it can be quite slow sometimes. Figure 5.1 describes how the test environment is built.

5.2 Scenarios

To find out the performance of the tracking filter and the different bias estimation methods a scenario has been simulated. The targets are assumed to travel on a straight line and in reality the measurements are supposed to have additive Gaussian noise. To become more realistic noise has been added to the the simulated radar measurements. The specific parameters are chosen to be $v_r \in \mathcal{N}(0, 100)$ (meters) and $v_\varphi \in \mathcal{N}(0, 0.01)$ (rad). To generate random values from a normal distribution the function `randn` in MATLAB has been used. The bias is uniformly chosen in the interval -5000 and 5000 meters.

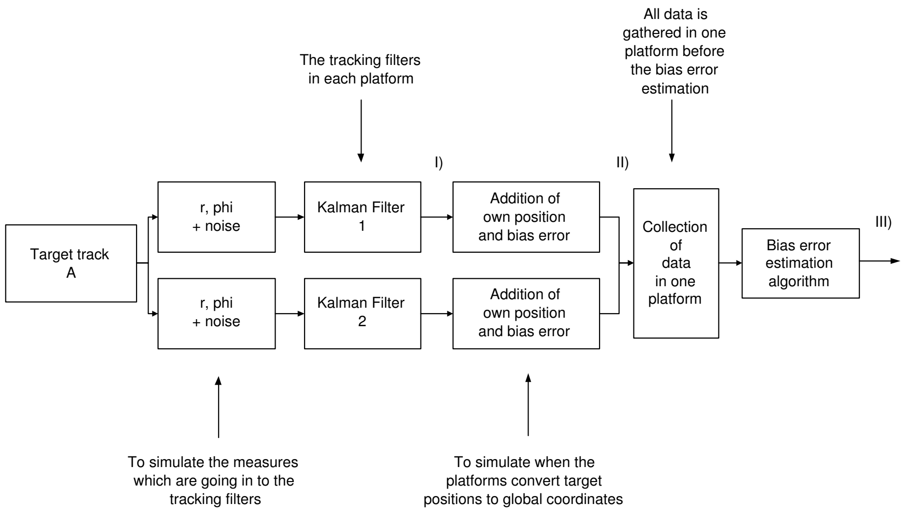


Figure 5.1. To simplify the situation only one target, named A, and two platforms, named 1 and 2, are considered. The model can easily be extended to more targets and platforms. The signals named I), II) and III) are the output state estimates from the tracking filters and the output from the bias filter. I) is target state estimates in local coordinates, $\hat{x}_A^1, P_A^1$ and $\hat{x}_A^2, P_A^2$. II) is target state estimates in global coordinates, $\hat{x}_A^1, P_A^1$ and $\hat{x}_A^2, P_A^2$. III) is the bias error state estimate, $\hat{\Delta}_A^{12}, P_A^{12}$.

5.3 Tracking Filter

As mentioned before there is a tracking filter in each platform and it is the data from the tracking filter that is used in the bias estimation process. It is interesting to see the performance of the tracking filter, see Section 4, because the bias error estimation process relies on the quality of the incoming data. Better quality of the data should give better result. In Figure 5.2 and Figure 5.3 the tracking process can be seen. The figures show the quality of the raw measurements and the estimated track. The plot is two dimensional and shows the position of the target in Cartesian coordinates. The scale of the axes are in meters and the track has been generated for 500 seconds, which gives us 500 data points.

5.4 Results

Results from Monte Carlo simulations for the different bias error estimation methods will be presented here. The results show the difference between the estimated bias compared to the real, here known, bias. As a measure of performance the root-mean-square-error ($RMSE$) is used, see for example [7]. A hundred Monte Carlo simulations are used. This was considered to be enough since no major change could be seen when more Monte Carlo simulations were performed.

To get good performance of the bias error estimation process the filters have

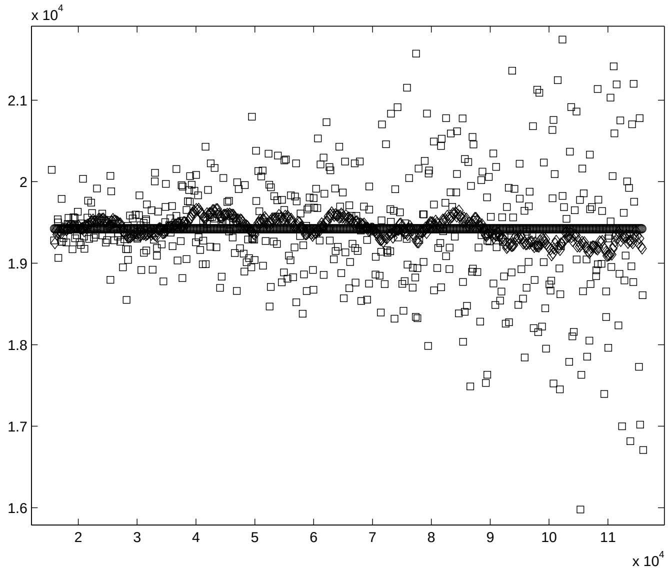


Figure 5.2. The tracking filter is estimating the state of a target. We see the raw measurements, squares, compared to the estimated track, diamonds, and the real generated track, circles (straight line). The unit is meters on both axes.

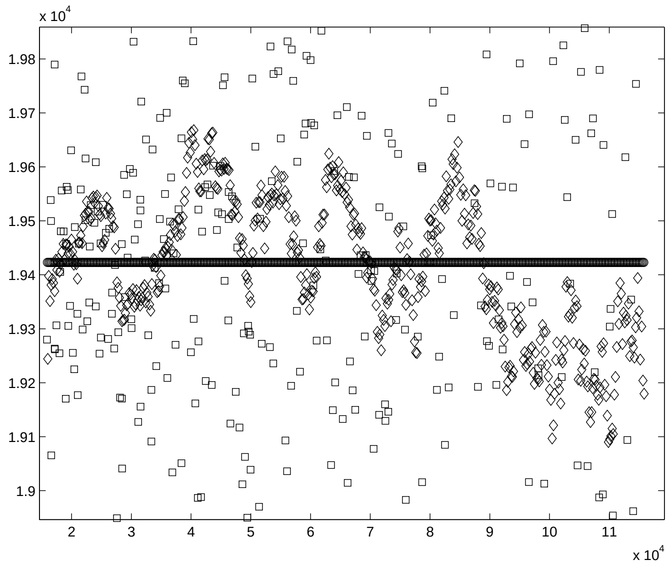


Figure 5.3. The tracking filter is estimating the state of a target. Figure 5.2 has been zoomed to see the track better. We see the raw measurements, squares, compared to the estimated track, diamonds, and the real generated track, circles (straight line). The unit is meters on both axes.

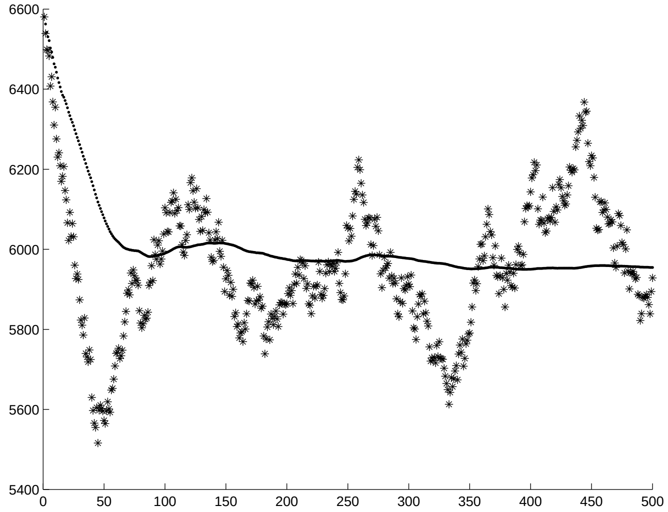


Figure 5.4. This plot compare the raw bias error, stars, with the filtered bias error, dots (the line). The figure is generated with the Kalman filter method and one common target is used. The unit of the x-axis is seconds and the y-axis meters.

been tuned. Nothing has been done to tune the tracking filter, it has been looked upon as a given system. In the other filters the variables; Q_Δ, R_Δ and P_0^Δ have been used as tuning parameters.

With the knowledge and aim that the filter should be fast but not to oscillatory trial and error with different combinations of the variables lead us to a final design.

We can compare the bias error before and after the estimation process, see Figure 5.4. The figure is generated with the Kalman filter method, but the result is very similar for the other methods as well.

5.4.1 Basic Kalman Filter

Results from the Kalman filter method, see Section 4.3, are presented here. In the first scenario there are two platforms estimating one common target. Figure 5.5 shows the result.

We can also see how the estimation is improved when there are five and ten targets in common. Figures 5.6(a) and 5.6(b) show the result.

5.4.2 Blackman Model

Results from the Blackman method, see Section 4.4, are presented here. In the first scenario there are two platforms and one common target. Figure 5.7 shows the result. The same simulation scenarios, as in the Kalman filter model, have been performed for the Blackman model.

We can also see how the estimation is improved when there are five and ten targets in common. Figures 5.8(a) and 5.8(b) show the result.

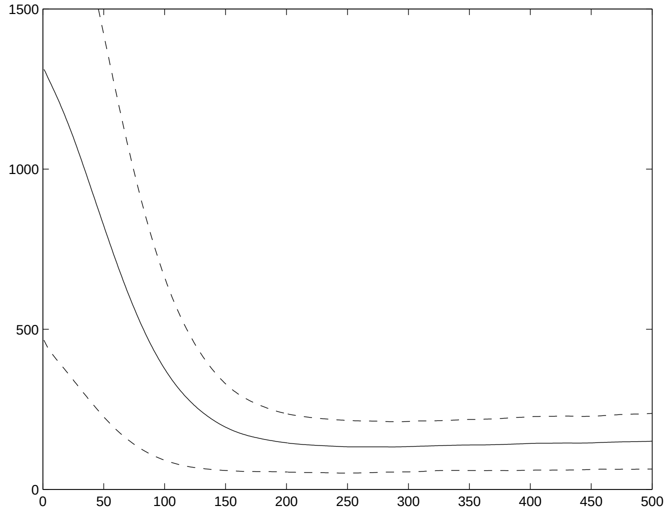
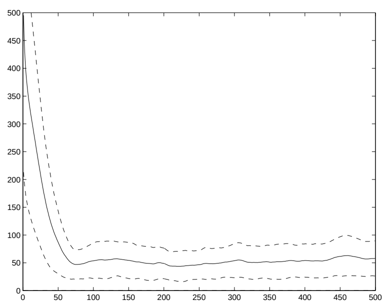
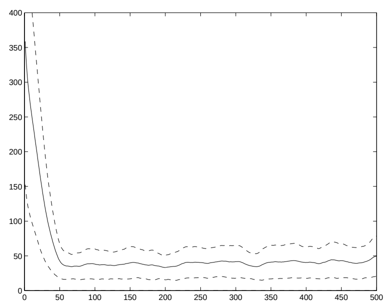


Figure 5.5. Kalman filter. The bias error is estimated and compared to the real bias error, the *RMSE* is the solid line. The two dashed lines show one standard deviation of the Monte Carlo simulations. The unit of the x-axis is seconds and the y-axis meters.



(a) Five common targets



(b) Ten common targets

Figure 5.6. Kalman filter. The bias error is estimated and compared to the real bias error, the *RMSE* is the solid line. The two dashed lines show one standard deviation of the Monte Carlo simulations. The unit of the x-axis is seconds and the y-axis meters.

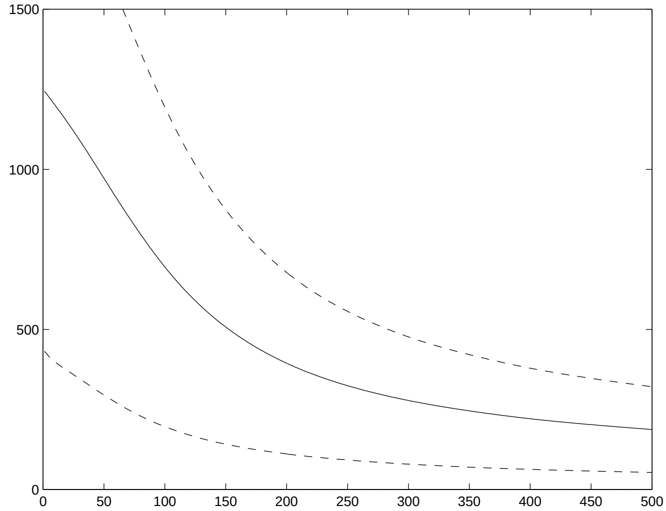
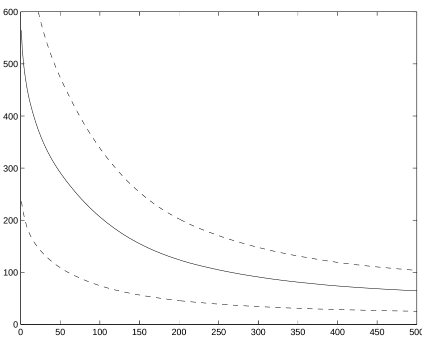
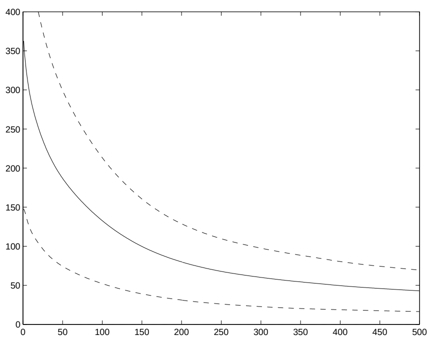


Figure 5.7. Blackman model. The bias error is estimated and compared to the real bias error, the *RMSE* is the solid line. The two dashed lines show one standard deviation of the Monte Carlo simulations. The unit of the x-axis is seconds and the y-axis meters.



(a) Five common targets



(b) Ten common targets

Figure 5.8. Blackman model. The bias error is estimated and compared to the real bias error, the *RMSE* is the solid line. The two dashed lines show one standard deviation of the Monte Carlo simulations. The unit of the x-axis is seconds and the y-axis meters.

5.4.3 Information Filter

Results from the information filter method, see Section 4.6, are presented here. In the first scenario there are two platforms and one common target. Figure 5.9 shows the result. The same simulation scenarios, as in the Kalman and Blackman filter models, have been used here. Additionally, to compare with the result gained if there would be access to the raw bias measurements, there is a least squares estimation of the raw bias error measurements. See Section 2.5.

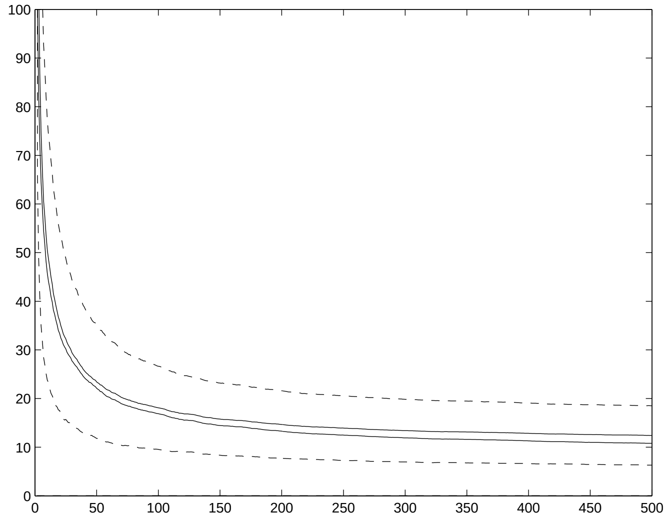
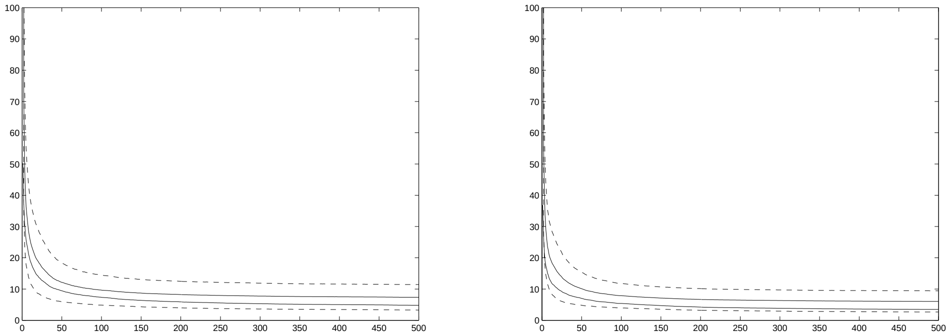


Figure 5.9. Information filter. The bias error is estimated and compared to the real bias error, the *RMSE* is the upper solid line. The lower solid line is the result from a recursive least squares estimation. The two dashed lines show one standard deviation of the Monte Carlo simulations. The unit of the x-axis is seconds and the y-axis meters.

We can also see how the estimation is improved when there are five and ten targets in common. Figures 5.10(a) and 5.10(b) show the result. There is also a zoomed version of Figure 5.10 since the estimate is becoming so good, see Figure 5.11.

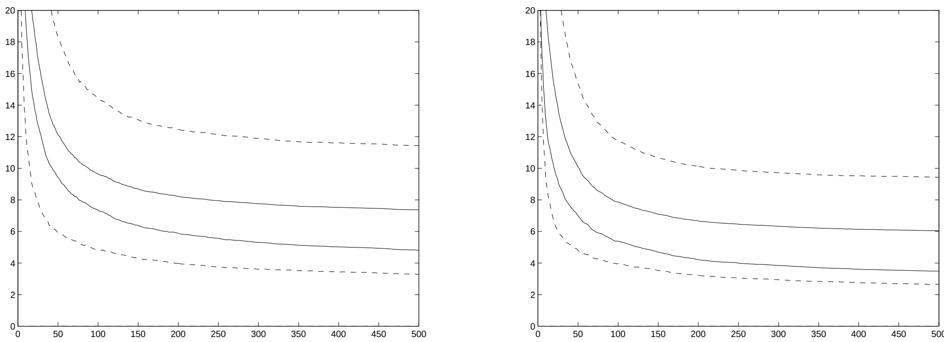
5.4.4 Computation Time

The algorithms require different amount of computational power. When choosing a specific algorithm performance could be the most important factor but never the less the algorithm has to work in the intended environment. Simulations using MATLABs built in functions `tic` and `toc` have been used to get the computation time for the different algorithms. Ten Monte Carlo simulations have been used to get a reasonable result. When performing the test several times similar results were obtained. The idea with the test is not to gain a very exact computation time but instead to get a notion of how the algorithms relate to each other. It can be seen that the Kalman filter method is the fastest one while the Blackman



(a) Five common targets (b) Ten common targets

Figure 5.10. Information filter. The bias error is estimated and compared to the real bias error, the *RMSE* is the upper solid line. The lower solid line is the result from a recursive least squares estimation. The two dashed lines show one standard deviation of the Monte Carlo simulations. The unit of the x-axis is seconds and the y-axis meters.



(a) Five common targets (b) Ten common targets

Figure 5.11. Information filter. A zoomed version of the bias error in Figure 5.10.

Table 5.1. Showing computation times (in seconds) for the three algorithms. Ten Monte Carlo simulations with different numbers of targets have been used.

Targets	Kalman filter	Blackman model	Information filter
1	9	16	28
5	33	64	88
10	63	122	162

model takes twice as much time and the information filter method is the slowest one and takes about three times as long time as the first one.

5.4.5 One Realization

The Monte Carlo simulations in previous sections have shown how the algorithms perform in average. It can also be interesting to see one snapshot, just one simulation, to get a notion of the performance. Figures 5.12, 5.13 and 5.14 show the result. Interesting results from this simulation is that the Blackman model is much smoother than the other two but on the other hand it is much slower. There should not be too much trust put in just one realization but the Kalman filter method seems to be more oscillatory than the information filter method which reaches a steady level after a longer period of time.

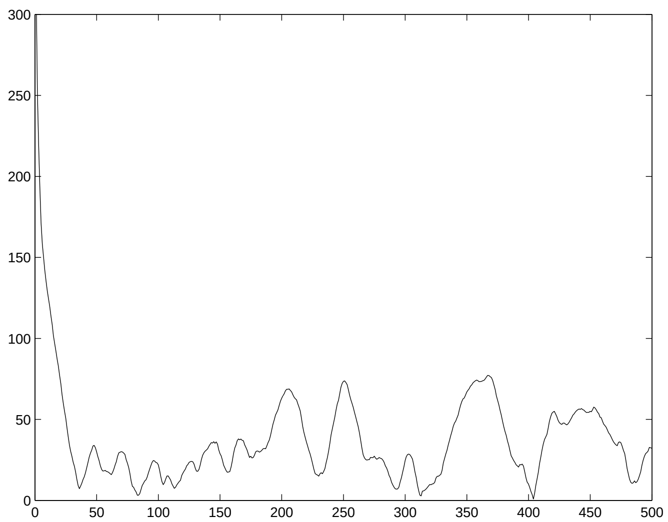


Figure 5.12. One realization of the Kalman filter method with five common targets, the RMSE is shown. The unit of the x-axis is seconds and the y-axis meters.

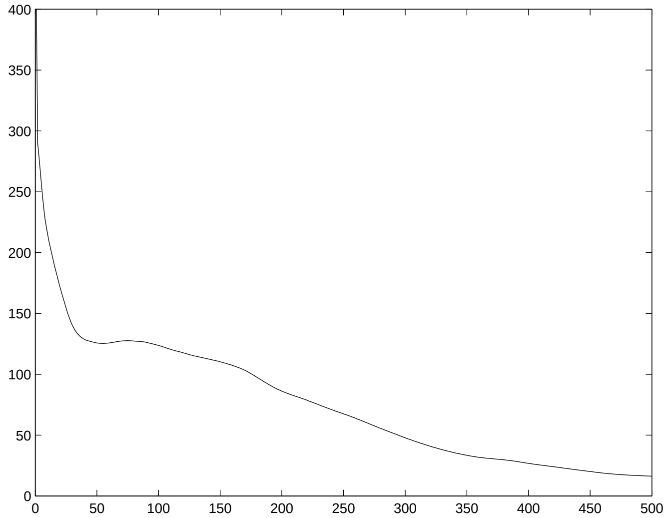


Figure 5.13. One realization of the Blackman filter method with five common targets, the RMSE is shown. The unit of the x-axis is seconds and the y-axis meters.

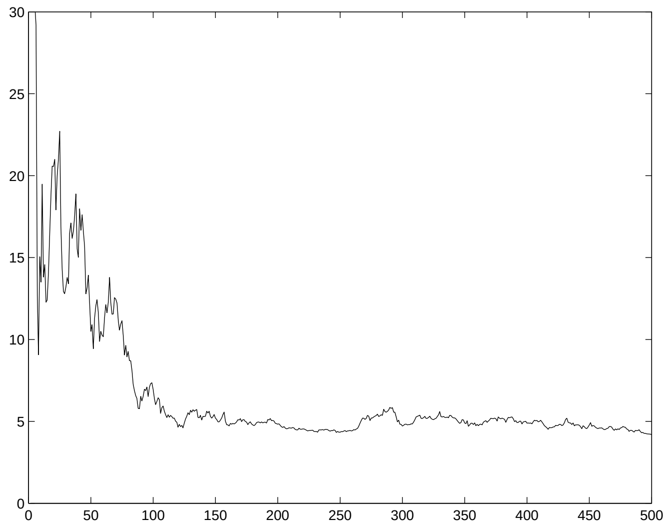


Figure 5.14. One realization of the information filter method with five common targets, the RMSE is shown. The unit of the x-axis is seconds and the y-axis meters.

Chapter 6

Conclusions

This chapter summarizes the work done and the results obtained. Conclusions will be drawn from the results and suggestions for further work and possible extensions will be made.

6.1 Discussion and Conclusions

In this master's thesis three bias error estimation algorithms have been developed. The purpose is to estimate the platform location bias so it can be compensated for and the data that is communicated via the data link can be corrected. A test environment has been built to simulate the tracking process and to estimate the bias error. The result is evaluated, see Chapter 5. Keep in mind the limitations (Section 1.3) made to the problem.

The first and second approach to bias error estimation are the Kalman filter model and the Blackman model respectively. They show rather similar results but the Kalman filter has a faster time response which is better since we want to estimate the target as fast as possible. The Blackman model has a greater standard deviation of the Monte Carlo simulations as well. The information filter gives a much better result than both the two mentioned algorithms. The reason to this could be that correlated information is not taken care of in the first two approaches. All three algorithms gives a better result when more common targets are available. If computing time is something to consider when choosing algorithm the information filter has to stand back in favor of the other two. Other limitations for example in hardware could also be crucial for the decision of method.

Considering some of the limitations made to the problem the result is better than it would be if for example there were more noise in the position of the platform or if the association was not perfect. On the other hand more sensors on a single platform would probably give better result.

As described in Section 1.3 perfect association is taken for granted. But if the association process is not so good as presumed and faulty measurements, apart from noise, appear in the raw bias measurements then what happens? If the association goes wrong not only one faulty measurement but several might appear.

Since the bias error is considered to be quite stationary it could be of interest to set some kind of threshold for the raw bias error measurements. If the measurements are not above the threshold or in between certain limits they are discarded to avoid the bias error estimation filter to include those measurements. This threshold is set to make the system more robust. Another solution is that bias error estimation could only be performed when no doubtful association scenarios appear.

On the other hand if a threshold is used in the filter from the beginning and assumes that the values in the beginning are correct it will never converge towards the correct bias error estimate. So somehow the filter would be more robust with a threshold but the threshold should only be applied later on in the process when the bias error estimate has come to a stationary level.

6.2 Further Work

During the work with this thesis the author has come across interesting issues to investigate further but due to time constraints things have had to be left out of this report. Suggestions for future development in this area is presented below.

- Implement the ad hoc method.

  When the ideas for the ad hoc method were developed another solution of the problem turned up, the information filter. Yet, it would be interesting to see how well the method would work. An advantage of this method compared to the information filter is that it requires less computational power. And if the application has high demands on fast computing this could for sure be a good alternative.

- Add a threshold for the raw bias error measurements.

  A threshold would probably make the process more robust towards faulty association and it would be interesting to see how well it could work out. Things that are of interest here is to evaluate how early one should apply the threshold and how large it should be. It could also be of interest to see what can go wrong with it. One hypothesis could be that the bias error for different targets should be about the same size.

- Association and gating.

  Once the bias error estimation algorithm has been developed it will affect the gating and association process. This would be interesting to investigate further and see how good performance it is possible to get. How should the gates be changed? And how does a bias error affects association.

Bibliography

[1] Ali T. Alouani, John E. Gray, and Hugh McCabe. Tracking with three asynchronous sensors: Solution existence and performance analysis. In Oliver E. Drummond, editor, *Proceedings of SPIE-Signal and Data Processing of Small Targets*, volume 4728, pages 368–376. SPIE, 2002.

[2] Yaakov Bar-Shalom, editor. *Multitarget-Multisensor Tracking, Advanced Applications*. Artech House, 1990. ISBN 0-89006-377-X.

[3] Samuel Blackman and Robert Popoli. *Design and analysis of modern tracking systems*. Artech House, 1999. ISBN 1-58053-006-0.

[4] Oliver E. Drummond. Distributed sensor registration. In Oliver E. Drummond, editor, *Proceedings of SPIE-Signal and Data Processing of Small Targets*, volume 4473, pages 323–333. SPIE, 2001.

[5] G. Girija and J.R. Raol. Application for information filter for sensor data fusion. *American Institute of Aeronautics and Astronautics*, AIAA-2000-0894, 2000.

[6] Neil Gordon and Branko Ristic. Performance bounds and algorithms for sensor registration in ecef coordinates. 2003. Intelligence, Surveillance and Reconnaissance Division.

[7] Fredrik Gustafsson. *Adaptive Filtering and Change Detection*. John Wiley & Sons, Ltd, 2000. ISBN 0-471-49287-6.

[8] Fredrik Gustafsson, Lennart Ljung, and Mille Millnert. *Signalbehandling*. Studentlitteratur, 2001. ISBN 91-44-01709-X.

[9] Gustaf Hendeby. Development and evaluation of an active radio frequency seeker model for a missile with data-link capability. Master's thesis, Linköpings universitet, 2002. LiTH-ISY-EX-3309.

[10] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME-Journal of Basic Engineering, 82(Series D):35-45*, March 1960.

[11] Rickard Karlsson. *Simulation Based Methods for Target Tracking*. Licentiate Thesis No. 930, Department of Electrical Engineering, Linköping Universitet, S-581 83 Linköping, Sweden, February 2002.

[12] Anders Malmberg. Målspårsassociation vid målföljning av flera mål med flera sensorer. Master's thesis, Linköpings universitet, 2000.

[13] Pär Norberg. Målföljning med decentraliserade K almanfilter. Master's thesis, Linköpings universitet, 1996. LiTH-ISY-EX-1575.

[14] Regina Rosander. Sensor fusion between a synthetic attitude and heading reference system and GPS. Master's thesis, Linköpings universitet, 2003. LiTH-ISY-EX-3408.

[15] Gregory A Watson and Theodore R Rice. Sensor alignment and compensation for composite tracking. In Oliver E Drummond, editor, *Proceedings of SPIE-Signal and Data Processing of Small Targets*, volume 4728, pages 354–367. SPIE, 2002.

Appendices

Appendix A

Notation

To simplify for the reader all notations used in the thesis is collected here.

Notation	Description
A	Linear state transition matrix, continuos.
A_t	Linear state transition matrix, discrete.
$\hat{a}_{t\|t}, \hat{a}_{t+1\|t}$	State vector in information filter.
B	Linear process noise propagation matrix, continuous.
B_t	Linear process noise propagation matrix, discrete.
C	Linear state transition matrix, continuous.
C_t	Linear state transition matrix, discrete.
$\frac{dh_t}{dx_t}$	Multidimensional derivative.
$E[\cdot]$	Expectancy operator.
$e_{\Delta t}$	Measurment noise in bias filter state equation.
H_t	Jacobian of mesurment function, discrete.
h_t	Non-linear measurment function, discrete.
I	Identity matrix.
K_t	Kalman gain matrix, discrete.
$P_{t\|t}, P_{t+1\|t}$	State covariance matrices, discrete.
$Q(t), \tilde{Q}(t)$	Process noise covariance matrix, continuous.
$Q_t, \tilde{Q}_t$	Process noise covariance matrix, discrete.
Q_Δ	Process noise covariance matrix for bias filter.
R_t	Measurment noise covariance matrix, discrete.
R_Δ	Measurment noise covariance matrix for bias filter.
T	Sampling time.
T_s	Sampling time.
x_0	Initial condition of state vector.
(x, y)	Cartesian coordinates.
$x(t)$	State, continuous.
$\dot{x}(t)$	Derivative of state, continuous.
x_t	State vector for tracking filter, discrete.
$\dot{x}_t$	Derivative of state, discrete.

Notation	Description
$\hat{x}_{t\|t}, \hat{x}_{t+1\|t}$	State estimation from the Kalman filter, discrete.
v_t	Measurment noise in the state space equation, discrete.
$\mathsf{v}_x, \mathsf{v}_y$	Velocities in cartesian coordinates.
v_r, v_φ	Measurment noise in tracking filter state space equation.
$w(t)$	Process noise in the state space equation, continuous.
w_t	Process noise in the state space equation, discrete.
Y_t	Measurment vector for information filter.
$y(t)$	Measurment, continuos.
y_t	Measurment, discrete.
z_t	Measurment vecor for bias filter.
$\hat{\Delta}$	Estimate of bias error state vector.
Δ_t	Bias error state vector.
δ_t	Dirac distribution centered around t.
σ_φ^2	Noise variance, $\sigma_\varphi^2 = E[v_\varphi v_\varphi^T]$.
σ_r^2	Noise variance, $\sigma_r^2 = E[v_r v_r^T]$.
Π_0	Initial condition of state covariance matrix.
Φ	State vector for information filter.

Appendix B

Covariance of Bias Error Estimate

The covariance matrix of the bias state estimate is derived as follows.

$$
\begin{aligned}
&E[\tilde{\Delta}^{12}\tilde{\Delta}^{12T}] = \\
&= E\big[(\hat{\Delta}^{12} - \Delta^{12})(\hat{\Delta}^{12} - \Delta^{12})^{T}\big] = \\
&E\left[\begin{pmatrix}\hat{\Delta}_x^{12} - \Delta_x^{12} \\ \hat{\Delta}_y^{12} - \Delta_y^{12}\end{pmatrix}\begin{pmatrix}\hat{\Delta}_x^{12} - \Delta_x^{12} & \hat{\Delta}_y^{12} - \Delta_y^{12}\end{pmatrix}\right] = \\
&E\left[\begin{pmatrix}(\hat{\Delta}_x^{12} - \Delta_x^{12})^2 & (\hat{\Delta}_x^{12} - \Delta_x^{12})(\hat{\Delta}_y^{12} - \Delta_y^{12}) \\ (\hat{\Delta}_y^{12} - \Delta_y^{12})(\hat{\Delta}_x^{12} - \Delta_x^{12}) & (\hat{\Delta}_y^{12} - \Delta_y^{12})^2\end{pmatrix}\right]
\end{aligned}
\tag{B.1}
$$

Now, chose to expand element $(1,1)$ which is the same expression as element $(2,2)$.

$$
\begin{aligned}
&E\Big[(\hat{\Delta}_x^{12} - \Delta_x{}^{12})^2\Big] = \\
&= E\Big[\Big((\hat{x}^1 - \hat{x}^2) - (x^1 - x^2)\Big)^2\Big] = \\
&= E\Big[\Big((\hat{x}^1 - x^1) - (\hat{x}^2 - x^2)\Big)^2\Big] = \\
&= E\Big[(\hat{x}^1 - x^1)^2 + (\hat{x}^2 - x^2)^2 - 2(\hat{x}^1 - x^1)(\hat{x}^2 - x^2)\Big] = \\
&= E\Big[(\hat{x}^1 - x^1)^2\Big] + E\Big[(\hat{x}^2 - x^2)^2\Big] - 2E\Big[(\hat{x}^1 - x^1)(\hat{x}^2 - x^2)\Big]
\end{aligned}
\tag{B.2}
$$

If $P^1 + P^2$ is used as an estimate for R_Δ and P_0^Δ only the terms

$$
E\Big[(\hat{x}^1 - x^1)^2\Big], \quad E\Big[(\hat{x}^2 - x^2)^2\Big]
$$

are included. The error made is done by neglecting the term

$$2E\Big[(\hat{x}^1 - x^1)(\hat{x}^2 - x^2)\Big]$$

which is hard to calculate.

Expand instead element $(1, 2)$ which is equivalent to element $(2, 1)$.

$$\begin{aligned}
E\Big[(\hat{\Delta}_x^{12} - {\Delta_x}^{12})(\hat{\Delta}_y^{12} - {\Delta_y}^{12})\Big] = \\
= E\Big[\Big((\hat{x}^1 - \hat{x}^2) - (x^1 - x^2)\Big)\Big((\hat{y}^1 - \hat{y}^2) - (y^1 - y^2)\Big)\Big] = \\
= E\Big[\Big((\hat{x}^1 - x^1) - (\hat{x}^2 - x^2)\Big)\Big((\hat{y}^1 - y^1) - (\hat{y}^2 - y^2)\Big)\Big] = \\
= E\Big[(\hat{x}^1 - x^1)(\hat{y}^1 - y^1) - (\hat{x}^1 - x^1)(\hat{y}^2 - y^2) - \\
(\hat{x}^2 - x^2)(\hat{y}^1 - y^1) - (\hat{x}^2 - x^2)(\hat{y}^2 - y^2)\Big] = \\
= E\Big[(\hat{x}^1 - x^1)(\hat{y}^1 - y^1)\Big] - E\Big[(\hat{x}^1 - x^1)(\hat{y}^2 - y^2)\Big] - \\
E\Big[(\hat{x}^2 - x^2)(\hat{y}^1 - y^1)\Big] - E\Big[(\hat{x}^2 - x^2)(\hat{y}^2 - y^2)\Big] \quad \text{(B.3)}
\end{aligned}$$

If $P^1 + P^2$ is used as an estimate for R_Δ and P_0^Δ only the terms

$$E\Big[(\hat{x}^1 - x^1)(\hat{y}^1 - y^1)\Big], \quad E\Big[(\hat{x}^2 - x^2)(\hat{y}^2 - y^2)\Big]$$

are included. Once again an error is made by neglecting the terms

$$E\Big[(\hat{x}^1 - x^1)(\hat{y}^2 - y^2)\Big], \quad E\Big[(\hat{x}^2 - x^2)(\hat{y}^1 - y^1)\Big]$$

which, again, are hard to determine.