

Applied particle filters in integrated aircraft navigation

Master's thesis
performed in **Automatic Control and**
Communication Systems
Performed for **Saab AB**

by
Petter Frykman

Reg no: LiTH-ISY-EX-3406-2003

28th April 2003

Applied particle filters in integrated aircraft navigation

Master's thesis

performed in **Automatic Control and
Communication Systems,**
Dept. of Electrical Engineering
at **Linköpings universitet**


Performed for **Saab AB**
by **Petter Frykman**

Reg no: LiTH-ISY-EX-3406-2003

Supervisor: **Per-Johan Nordlund**
Saab AB
Thomas Schön
Linköpings universitet

Examiner: **Professor Fredrik Gustafsson**
Linköpings universitet

Linköping, 28th April 2003

		Avdelning, Institution Division, Department Automatic Control and Communication Systems, Dept. of Electrical Engineering 581 83 Linköping	Datum Date 28th April 2003
Språk Language <input type="checkbox"/> Svenska/Swedish <input checked="" type="checkbox"/> Engelska/English <input type="checkbox"/> _____	Rapporttyp Report category <input type="checkbox"/> Licentiatavhandling <input checked="" type="checkbox"/> Examensarbete <input type="checkbox"/> C-uppsats <input type="checkbox"/> D-uppsats <input type="checkbox"/> Övrig rapport <input type="checkbox"/> _____	ISBN —	
URL för elektronisk version http://www.control.isy.liu.se http://www.ep.liu.se/exjobb/isy/2003/3406/		ISRN LITH-ISK-EX-3406-2003 Serietitel och serienummer ISSN Title of series, numbering —	
Titel Tillämpning av partickelfilter i integrerad flygplansnavigering Title Applied particle filters in integrated aircraft navigation Författare Petter Frykman Author			
Sammanfattning Abstract <p>Navigation is about knowing your own position, orientation and velocity relative to some geographic entities. The sensor fusion considered in this thesis combines data from a dead reckoning system, inertial navigation system (INS), and measurements of the ground elevation. The very fast dynamics of aircraft navigation makes it difficult to estimate the true states. Instead the algorithm studied will estimate the errors of the INS and compensate for them. A height database is used along with the measurements. The height database is highly non-linear why a Rao-Blackwellized particle filter is used for the sensor fusion. This integrated navigation system only uses data from its own sensors and from the height database, which means that it is independent of information from outside the aircraft.</p> <p>This report will describe the algorithm and illustrate the theory used. The main purpose is to evaluate the algorithm using real flight data, why the result chapter is the most important.</p>			
Nyckelord Integrated navigation, Inertial navigation, Aircraft navigation, particle filter, Multiple model, Monte Carlo filter, Rao-Blackwellization Keywords			

Abstract

Navigation is about knowing your own position, orientation and velocity relative to some geographic entities. The sensor fusion considered in this thesis combines data from a dead reckoning system, inertial navigation system (INS), and measurements of the ground elevation. The very fast dynamics of aircraft navigation makes it difficult to estimate the true states. Instead the algorithm studied will estimate the errors of the INS and compensate for them. A height database is used along with the measurements. The height database is highly non-linear why a Rao-Blackwellized particle filter is used for the sensor fusion. This integrated navigation system only uses data from its own sensors and from the height database, which means that it is independent of information from outside the aircraft.

This report will describe the algorithm and illustrate the theory used. The main purpose is to evaluate the algorithm using real flight data, why the result chapter is the most important.

Keywords: Integrated navigation, Inertial navigation, Aircraft navigation, particle filter, Multiple model, Monte Carlo filter, Rao-Blackwellization

Preface

Acknowledgments

I never thought that so many people would be involved when I began this thesis. First of all I would like to express my deepest gratitude to my two supervisors, Per-Johan Nordlund at Saab who have helped me grasp the theory and Thomas Schön at Linköping universitet who have guided me in the jungle of writing the report. I would also like to thank my examiner professor Fredrik Gustafsson at Linköpings universitet who have made me realize that the thesis is not supposed to be a course reader and therefore do not need the general theory but should instead enlighten the ideas.

My opponent Stefan Johnsson and also Daniel Frylmark have given me a great deal of help in writing the thesis at a suitable level. Without their comments lots of things that is obvious to me would not have been explained more closely. Regina Rosander, who has been writing her master's thesis at Saab Aerospace AB at the same time as I, has also contributed with comments about the language and the technical level of the thesis.

The people at the department of Primary data and Navigation at Saab Aerospace AB have all been very helpful and given me information about the navigation system of today.

Thank you all!

Linköping, 28th April 2003

Petter Frykman

Contents

Abstract	v
Acknowledgments	vii
1 Introduction	1
1.1 Background	1
1.2 Specification	2
1.3 Thesis outline	2
2 Recursive Bayesian estimation	3
2.1 Kalman filter	4
2.2 Multiple model	5
2.3 Particle filter	7
2.3.1 Importance sampling	7
2.3.2 Resampling	10
2.3.3 The algorithm	11
2.4 Rao-Blackwellization	11
2.4.1 Rao-Blackwellized filter	12
2.5 Cramer-Rao bounds	15
3 Integrated aircraft navigation	17
3.1 Aircraft navigation	17
3.2 Inertial navigation	21
3.3 Terrain aided positioning	22
3.4 Error dynamic model	23
3.4.1 Position error dynamics	25
3.4.2 Velocity error dynamics	26
3.4.3 Altitude error dynamics	28
3.4.4 Attitude error dynamics	29
3.4.5 Sensor noise	29
3.4.6 Continuous state space equations	29
3.4.7 Discrete state space equations	30
3.4.8 Summary	30

4	Sensor fusion	31
4.1	Problem decomposition	31
4.2	Horizontal position error	32
4.3	Altitude error	32
4.4	Conditionally linear part	33
4.5	Sensor fusion	34
5	Simulations	39
5.1	Flights	39
5.2	Parameters	40
5.2.1	Number of samples	40
5.2.2	Initial values	41
5.2.3	Process noise	43
5.2.4	Measurement noise	43
5.2.5	Parameter summary	45
5.3	Evaluation methods	45
5.4	Constant map covariance	46
5.4.1	Results	46
5.5	Variable map covariance	50
5.5.1	Results	50
5.6	Variable added process noise	51
5.6.1	Cramer-Rao bounds	52
5.6.2	Results	52
5.7	Comparison to existing methods	53
6	Conclusions and future work	57
6.1	Conclusion	57
6.2	Future work	58
	References	59
	Notation	61
A	Continuous state space model	63

Chapter 1

Introduction

1.1 Background

There are many different ways of navigation and this thesis will evaluate one that is based only on knowledge that the aircraft is carrying itself. This system is a more technically advanced version of the techniques used in hundreds of years by sailors, dead reckoning and observations. If the initial position is known you can easily calculate the current position if you have access to speed, heading and time. Errors in the measurements introduce errors in the calculations that are accumulated over time. Every now and then the sailor makes observations of his own position, for example by observing the bearing of two lighthouses. These observations can be used to compensate for the errors in the calculations and also give information about error characteristics. If the errors are systematic they are probably due to the current and he can compensate for them in future calculations.

Now there are sensors and algorithms that take care of this. Instead of using velocity and heading, the system uses acceleration and angular velocities since they can be measured with good precision by sensors onboard the aircraft. These can be integrated to obtain the position, velocity and heading. This system is called Inertial Navigation System (INS) and is used in most aircraft navigation systems of today. The problem is that they slowly drift from the real states and therefore have to be stabilized, just as the sailor needed to make observations. One sensor often used is the Global Positioning System (GPS). A system that uses data from one sensor, like GPS, to stabilize a navigation system, like INS, is called an integrated navigation system.

In military airforce applications you often try to develop navigation systems that are independent of information from outside the aircraft. Information from the GPS satellites for example may be corrupt or

jammed. One of the more recent methods developed is Terrain Aided Positioning (TAP). In TAP the height profile of the ground below is used to find the position. To do this, the aircraft must have a height database. This database is highly non-linear and the sensor fusion of the position data from TAP and data from other navigation systems is therefore very complex.

1.2 Specification

Saab Aerospace AB has been developing the Terrain Aided Positioning system for quite some time and has a system called New Integrated Navigation System (NINS) running today. This system is based on the INS and the TAP. The sensor fusion is carried out by an Extended Kalman filter using the data from INS and the position data from TAP. Since the Kalman filter is a linear filter the sensor fusion is linearized completely.

A new sensor fusion, based on Monte Carlo simulations, has been developed. By using non-linear filtering techniques the problem does not have to be completely linearized and the performance should therefore be better. This thesis will evaluate this new sensor fusion using real flight data and also investigate some of the problems with missing data, i.e., when there are no measurements that the algorithm can use.

For me this thesis is the last part of my master of science education in Electrical Engineering at Linköpings universitet. The work was done from October 2002 to March 2003 at Saab Aerospace AB in Linköping.

1.3 Thesis outline

Chapter 2 will cover the theory in a very brief manner and can be skipped the first time. In fact, it is more an intuitive motivation of how the filters work, see Nordlund [14] for a more thorough investigation. In Chapter 3 the equations for aircraft navigation are presented along with some background theory. Chapter 4 describes how the theory from Chapter 2 and the equations from Chapter 3 fit together. Chapter 5 presents the results of the algorithm, run on real flight data, and finally Chapter 6 will discuss some conclusions and future work.

Chapter 2

Recursive Bayesian estimation

In many applications the goal is to estimate the posterior probability density for the states by making some observations. Consider the non-linear state space model

$$x_{t+1} = f_t(x_t) + G_t u_t \quad (2.1a)$$

$$y_t = h_t(x_t) + e_t, \quad (2.1b)$$

where x_t is the state vector, y_t is the observation, u_t is the process noise and e_t is the measurement noise. Also let $X_t = \{x_0, x_1, \dots, x_t\}$ and $Y_t = \{y_0, y_1, \dots, y_t\}$ be the stacked vectors of states and observations up to time t . Assume that u_t and e_t are both independent and has known density functions. In this case the state of the system is a Markov process

$$p(X_t) = p(x_0) \prod_{k=1}^t p(x_k | x_{k-1}),$$

and the measurements, Y_t , are conditionally independent given the states, X_t

$$p(Y_t | X_t) = \prod_{k=0}^t p(y_k | x_k).$$

Obviously the size of these expressions grows as time evolves if we were to calculate everything from scratch. To be able to estimate the posterior in real time, we need a way to use the estimation that we have at time $t - 1$ to calculate the estimation at time t . The following

recursive equations are used

$$p(x_{t+1}|Y_t) = \int p(x_{t+1}|x_t) p(x_t|Y_t) dx_t \quad (2.2a)$$

$$p(x_{t+1}|Y_{t+1}) = \frac{p(y_{t+1}|x_{t+1}) p(x_{t+1}|Y_t)}{p(y_{t+1}|Y_t)}. \quad (2.2b)$$

The first equation is called the time update equation and the second is called the measurement update equation.

Often there does not exist closed-form expressions for $p(x_t|Y_t)$. Depending on the characteristics of the system, there exist different methods of estimating $p(x_t|Y_t)$. In this thesis there will be three different cases of filtering, namely linear Gaussian, linear Gaussian mixture, and non-linear. Methods for these three cases are described in the following three sections. Finally, a description of how to combine these methods is presented.

2.1 Kalman filter

When the model is linear and Gaussian, the optimal recursive estimate is given by the Kalman filter. Consider the linear model

$$x_{t+1} = F_t x_t + G_t u_t \quad (2.3a)$$

$$y_t = H_t x_t + e_t, \quad (2.3b)$$

where

$$u_t \sim \mathcal{N}(0, Q_t), \quad e_t \sim \mathcal{N}(0, R_t), \quad x_0 \sim \mathcal{N}(\hat{x}_0, P_0), \quad (2.4)$$

i.e., white independent noise. Kalman tried to find a least squares error estimation of the state vector given the observations. He showed that when Y_t is given, x_t and x_{t+1} are Gaussian distributed according to

$$p(x_t|Y_t) = \mathcal{N}(\hat{x}_{t|t}, P_{t|t}) \quad (2.5a)$$

$$p(x_{t+1}|Y_t) = \mathcal{N}(\hat{x}_{t+1|t}, P_{t+1|t}), \quad (2.5b)$$

where

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + P_{t|t-1} H_t^T S_t^{-1} (y_t - H_t \hat{x}_{t|t-1}) \quad (2.6a)$$

$$P_{t|t} = P_{t|t-1} - P_{t|t-1} H_t^T S_t^{-1} H_t P_{t|t-1} \quad (2.6b)$$

$$S_t = R_t + H_t P_{t|t-1} H_t^T \quad (2.6c)$$

$$\hat{x}_{t+1|t} = F_t \hat{x}_{t|t} \quad (2.6d)$$

$$P_{t+1|t} = F_t P_{t|t} F_t^T + G_t Q_t G_t^T, \quad (2.6e)$$

with initial values $\hat{x}_{0|-1} = \hat{x}_0$ and $P_{0|-1} = P_0$. In e.g., Gustafsson [8] a formal proof can be found.

2.2 Multiple model

In this thesis the observations will have a non-Gaussian measurement noise. Instead we can estimate it by two Gaussians, more details will be presented in Section 4.3. There is a framework, called multiple model, that deals with this kind of problems. In this section a simplified version of the Generalized Pseudo Bayesian (GPB) algorithm is presented. For a more detailed description of the GPB see e.g., Nordlund [13, 14]. There is a similar method called interacting multiple model see e.g., Blom Bar-Shalom [3].

Consider the linear model

$$x_{t+1} = F_t x_t + G_t u_t \quad (2.7a)$$

$$y_t = H_t x_t + e_t(\lambda_t), \quad (2.7b)$$

where $\lambda_t = \{1, 2\}$ corresponds to the mode of the system and

$$e_t(\lambda_t) = \begin{cases} \mathcal{N}(m_1, \sigma_1), & \lambda_t = 1 \\ \mathcal{N}(m_2, \sigma_2), & \lambda_t = 2, \end{cases} \quad (2.8)$$

are either of the two Gaussians. The mode is stochastic and is often considered to be a Markov process. In this thesis the mode is independent. If the mode history $\Lambda_t = \{\lambda_0, \dots, \lambda_t\}$ is known, the model is Gaussian and a Kalman filter can be used for the estimation problem. The law of total probability gives

$$p(x_t|Y_t) = \sum_{\Lambda_t} p(\Lambda_t|Y_t) p(x_t|Y_t, \Lambda_t), \quad (2.9)$$

where a Kalman filter gives $p(x_t|Y_t, \Lambda_t)$. The sum is over all possible mode histories, and is hence an exponentially growing sum. The obvious way to handle this problem is to only consider the L latest modes

$$p(x_t|Y_t) \approx \sum_{\Lambda_{t-L+1}^t} p(\Lambda_{t-L+1}^t|Y_t) p(x_t|Y_t, \Lambda_{t-L+1}^t), \quad (2.10)$$

where $\Lambda_{t-L+1}^t = \{\lambda_{t-L+1}, \dots, \lambda_t\}$. The approximation here is to consider $p(x_t|Y_t, \Lambda_{t-L+1}^t)$ as a Gaussian. This method is called the Generalized Pseudo Bayesian (GPB), and the notation GPB1 is used when $L = 1$. GPB1 starts with an estimate of the state and covariance according to a Kalman filter. For each time step it runs two Kalman filters corresponding to the two possible modes, tries to estimate the probability for each mode and merges the two according to their probabilities. Figure 2.1 illustrates the GPB1 algorithm. In the figure there is a new notation for the probability $p(\lambda_t|Y_t)$, namely the weights $\bar{\alpha}_t$.

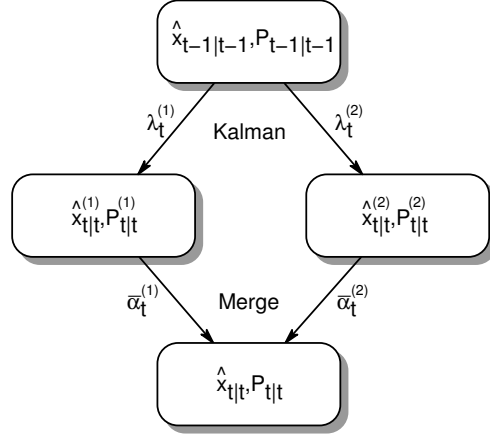


Figure 2.1: The GPB1 recursive structure.

This comes from the fact that they can only be estimated relatively and then be normalized, hence the name weights. Repeated use of Bayes' rule gives

$$p(\lambda_t | Y_t) = \frac{p(y_t | \lambda_t, Y_{t-1}) p(\lambda_t | Y_{t-1})}{p(Y_t | Y_{t-1})} = \frac{p(y_t | \lambda_t, Y_{t-1}) p(\lambda_t)}{p(Y_t | Y_{t-1})}, \quad (2.11)$$

where the last equality comes from the fact that the mode is an independent stochastic process. Since the denominator will be the same for every mode, it is enough to calculate the numerator and normalize the result

$$\alpha_t^{(i)} = p(y_t | \lambda_t = i, Y_{t-1}) p(\lambda_t = i) \quad (2.12a)$$

$$\bar{\alpha}_t^{(i)} = \frac{\alpha_t^{(i)}}{\sum_j \alpha_t^{(j)}}. \quad (2.12b)$$

In the sensor fusion in Chapter 4 the GPB1 will be used. Algorithm 2.1 is a summary of this algorithm.

Algorithm 2.1. (GPB1)

1. Time update

Calculate $\hat{x}_{t|t-1}$ and $P_{t|t-1}$ according to a Kalman filter.

2. Measurement update, split

Calculate $\hat{x}_{t|t}^{(1)}$, $P_{t|t}^{(1)}$ using a Kalman filter for $\lambda_t = 1$ and $\hat{x}_{t|t}^{(2)}$, $P_{t|t}^{(2)}$ using a Kalman filter for $\lambda_t = 2$.

3. Weights

Calculate $\bar{\alpha}_t^{(k)}$ from

$$p(y_t | \lambda_t = k) = \mathcal{N}(y_t - H_t \hat{x}_{t|t-1} - m_k, \sigma_k + H_t P_{t|t-1} H_t^T),$$

$$k = \{1, 2\}.$$

4. Merge

Calculate $\hat{x}_{t|t} = \sum_{k=1}^2 \bar{\alpha}_t^{(k)} \hat{x}_{t|t}^{(k)}$ and

$$P_{t|t} = \sum_{k=1}^2 \bar{\alpha}_t^{(k)} \left(P_{t|t}^{(k)} + \left(\hat{x}_{t|t}^{(k)} - \hat{x}_{t|t} \right) \left(\hat{x}_{t|t}^{(k)} - \hat{x}_{t|t} \right)^T \right).$$

2.3 Particle filter

One method to handle non-linear and non-Gaussian filtering problems is to use particle filters, also referred to as Monte Carlo filters. A particle filter can be seen as a huge number of simulations. These will be referred to as samples in this thesis, particles is another common notation found in literature. Each sample has some kind of weight associated to it, indicating how well its states agree to the measurements. The samples and their corresponding weights are combined to form an estimate of the desired posterior. For every time step the samples are more and more likely to drift away from the real state meaning that most of the weights will tend to zero. To prevent this from happening the samples with smaller weights will die and those with larger weights will multiply in some kind of evolution process. Below is a presentation of the theory in a very brief manner and totally without proofs. For a more thorough investigation of particle filters see e.g., Andrieu et al. [1] or Nordlund [12, 13].

2.3.1 Importance sampling

Consider the non-linear model

$$x_{t+1} = f(x_t) + u_t \tag{2.13a}$$

$$y_t = h(x_t) + e_t, \tag{2.13b}$$

where the process and measurement noises, $u_t \sim p_{u_t}(\cdot)$ and $e_t \sim p_{e_t}(\cdot)$ respectively, are assumed independent with known densities. The state of the system is in this case a Markov process

$$p(X_t) = p(x_0) \prod_{k=1}^t p(x_k | x_{k-1}),$$

where the prior distribution of the state at time $t = 0$ is given by $p(x_0)$. The observations are conditionally independent given the states

$$p(Y_t|X_t) = \prod_{k=0}^t p(y_k|x_k).$$

As before we like to estimate the posterior density $p(x_t|Y_t)$. Since the model is neither linear nor Gaussian, the posterior cannot be represented in any other way than the total probability density function. Recursive Monte Carlo simulations give an estimate of $p(X_t|Y_t)$. Let us consider the recursive equation

$$\begin{aligned} p(X_t|Y_t) &= \frac{p(y_t|X_t, Y_{t-1}) p(x_t|X_{t-1}, Y_{t-1})}{p(y_t|Y_{t-1})} p(X_{t-1}|Y_{t-1}) \\ &= \frac{p(y_t|x_t) p(x_t|x_{t-1})}{p(y_t|Y_{t-1})} p(X_{t-1}|Y_{t-1}), \end{aligned} \quad (2.14)$$

where the last equality follows from the fact that the system is a Markov process as stated above.

The particle filter estimates the density function as a discretized version by utilizing a large number of samples. Let $\{X_t^{(i)}\}_{i=1}^N$ be samples drawn from the posterior. Then the expression for the estimate of the posterior is

$$\hat{P}(X_t|Y_t) = \frac{1}{N} \sum_{i=1}^N \delta(X_t - X_t^{(i)}), \quad (2.15)$$

where $\delta(X_t)$ is the Dirac delta function. In this case all samples are equally correct as samples from the posterior, since they are drawn from the posterior itself. Therefore their weights in the sum is equal and can be set to one. In order to satisfy the law of total probability, the estimate has to be multiplied with $1/N$. How good the estimate is depends on the number of samples.

This estimate can be used to calculate different moments of the posterior, for example the expectation and covariance, according to

$$\begin{aligned} E_{p(x_t|Y_t)}[\cdot] &\approx E_{p(X_t|Y_t)}[\cdot] \\ \hat{x} &= E_{p(x_t|Y_t)}[x] = \int p(x) x dx \\ &\approx \int \frac{1}{N} \sum_{i=1}^N \delta(x - x^{(i)}) x dx = \frac{1}{N} \sum_{i=1}^N x^{(i)} \end{aligned} \quad (2.16a)$$

$$\begin{aligned}
P &\approx \int \frac{1}{N} \sum_{i=1}^N \delta(x - x^{(i)}) (x - \hat{x}) (x - \hat{x})^T dx \\
&= \frac{1}{N} \sum_{i=1}^N (x^{(i)} - \hat{x}) (x^{(i)} - \hat{x})^T.
\end{aligned} \tag{2.16b}$$

Now the samples cannot be drawn from the posterior since it is unknown. Instead they are drawn from a known probability density $q(X_t|Y_t)$. Using Bayes' rule gives

$$\begin{aligned}
q(X_t|Y_t) &= q(x_t|X_{t-1}, Y_t) q(X_{t-1}|Y_t) \\
&= q(x_t|X_{t-1}, Y_t) q(X_{t-1}|Y_{t-1}),
\end{aligned} \tag{2.17}$$

where the last equality is the result of introducing the restriction that the states at time $t-1$ and older are independent of the measurement at time t . This means that we can draw $\{x_t^{(i)}\}_{i=1}^N$ from $q(x_t|X_{t-1}, Y_t)$ and form the set $\{X_t^{(i)} = \{X_{t-1}^{(i)}, x_t^{(i)}\}\}_{i=1}^N$ without adjusting $\{X_{t-1}^{(i)}\}_{i=1}^N$.

To use these samples to estimate the posterior each sample is associated with the so called importance weight

$$w_t^{(i)} = \frac{p(X_t^{(i)}|Y_t)}{q(X_t^{(i)}|Y_t)} = c_t \frac{p(y_t|x_t^{(i)}) p(x_t^{(i)}|x_{t-1}^{(i)})}{q(x_t^{(i)}|X_{t-1}^{(i)}, Y_t)} w_{t-1}^{(i)}, \tag{2.18}$$

where $c_t = p(Y_{t-1})/p(Y_t)$. Only the relative relationship between the weights are important and c_t can therefore be neglected. This gives the weight update equation

$$w_t^{(i)} = \frac{p(y_t|x_t^{(i)}) p(x_t^{(i)}|x_{t-1}^{(i)})}{q(x_t^{(i)}|X_{t-1}^{(i)}, Y_t)} w_{t-1}^{(i)}. \tag{2.19}$$

A simple, but effective choice is to draw from the state propagation density, i.e.,

$$q(x_t|X_{t-1}, Y_t) = p(x_t|x_{t-1}).$$

The corresponding weight update equation will be

$$w_t^{(i)} = p(y_t|x_t^{(i)}) w_{t-1}^{(i)}. \tag{2.20}$$

With the samples drawn from $q(x_t|X_{t-1}, Y_t)$ along with the importance weights the new estimate of the posterior $p(X_t|Y_t)$ is

$$\hat{P}(X_t|Y_t) = \sum_{i=1}^N \bar{w}_t^{(i)} \delta(X_t - X_t^{(i)}) \tag{2.21a}$$

$$\bar{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^N w_t^{(j)}}, \tag{2.21b}$$

and we can use this fact to get an estimate of the current state, x_t , according to

$$\hat{x}_{t|t} = E_{p(X_t|Y_t)} [x_t] \approx \sum_{i=1}^N \bar{w}_t^{(i)} x_t^{(i)}, \quad (2.22)$$

and the covariance

$$\begin{aligned} P_{t|t} &= E_{p(X_t|Y_t)} \left[(x_t - E_{p(X_t|Y_t)} [x_t])^2 \right] \\ &\approx \sum_{i=1}^N \bar{w}_t^{(i)} \left(x_t^{(i)} - \hat{x}_{t|t} \right) \left(x_t^{(i)} - \hat{x}_{t|t} \right)^T. \end{aligned} \quad (2.23)$$

2.3.2 Resampling

As time evolves the samples tend to spread and the weights will be almost zero for most of the samples, meaning that they do not contribute much to the estimation of the posterior. It also means that the estimate in the interesting region is crude, since there are not many samples contributing. There is a way to know when this happens, and the problem can even be solved.

Effective sample size is a way to measure how well the samples are concentrated in the interesting region. By comparing the covariance of a set of samples drawn from the posterior and the covariance obtained through the use of importance sampling we will get a measurement of the sampling efficiency. This in turn will give an expression for the effective sample size. In e.g., Nordlund [14] it is shown that the effective sample size can be estimated by

$$\hat{N}_{\text{eff}} \approx \frac{1}{\sum_i \left(\bar{w}_t^{(i)} \right)^2}. \quad (2.24)$$

If all the weights are equal, the effective sample size will be N . One way to decide when the samples have spread far enough is to use a lower threshold for the effective sample size. Later we will use $N_{\text{th}} = 2N/3$ as the threshold.

When the samples move away from the real state their weights decrease. This in turn will decrease the effective sample size which eventually will pass the threshold. When this happens we draw N new samples from $\{X_t^{(i)}\}$ with replacement, where the probability of choosing $X_t^{(i)}$ is $\bar{w}_t^{(i)}$. The new set of samples are drawn from the estimate of the posterior and all the weights should therefore be set to $1/N$. By doing this, the samples with low weights will be discarded and the samples with high weights will be multiplied. In this way the cloud of samples is concentrated to the interesting region.

There is a price to pay for this solution; the samples are no longer independent because some of them will be copies of the same sample. As long as we do not resample too often, this problem turns out to be of less importance. In this thesis a minimum number of time steps between resampling is used.

2.3.3 The algorithm

The theory described in this chapter is only to be considered as a motivation of how particle filters work. It has been presented in a way that is suitable for the application in this thesis. Algorithm 2.2 is a summary of the particle filter used. It starts by sampling samples from the prior distribution and calculates the first set of weights from the first measurement. In each iteration the samples are propagated only according to the model. The measurement is only used when updating the weights. If another distribution for sampling is used this fact may not be true.

Algorithm 2.2 (The particle filter).

At time $t = 0$:

1. For $i = 1, \dots, N$, sample $x_0^{(i)} \sim p(x_0)$.
2. For $i = 1, \dots, N$, calculate $w_0^{(i)} = p(y_0|x_0^{(i)})$
and normalize $\bar{w}_0^{(i)} = \frac{w_0^{(i)}}{\sum_j w_0^{(j)}}$.

For each time $t \geq 1$:

1. If resampling (e.g., $N_{\text{eff}} < N_{\text{th}}$), then resample
and set the weights $\bar{w}_{t-1}^{(i)} = \frac{1}{N}$.
2. For $i = 1, \dots, N$, sample $x_t^{(i)} \sim p(x_t|x_{t-1}^{(i)})$.
3. For $i = 1, \dots, N$, update the weights $w_t^{(i)} = p(y_t|x_t^{(i)}) \bar{w}_{t-1}^{(i)}$
and normalize $\bar{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_j w_t^{(j)}}$.

2.4 Rao-Blackwellization

From the central limit theorem, it can be shown that the estimation error is independent of the dimensionality of the problem. However, it turns out that the dimension has influence on the number of samples

needed. Tests show that the same number of samples will yield a better accuracy for a problem of low dimension than for one of high. The higher dimensionality, the more samples needed to cover the state space efficiently is an intuitive explanation. In some problems, there is a structure in the state space formulation. This may be used to split the problem into one part that can be estimated in a closed form, and leave the other part to simulation based methods. By using this structure, the accuracy increases while using the same number of samples. Below follows a description of the Rao-Blackwellization algorithm in a way that is suitable for this thesis. For more theory see e.g., Chen Liu [4], Doucet et al. [6] and Doucet et al. [5].

2.4.1 Rao-Blackwellized filter

Consider the model

$$x_{t+1}^p = f_t^p(x_t^p) + F_t^p(x_t^p)x_t^k + G_t^p(x_t^p)u_t^p \quad (2.25a)$$

$$x_{t+1}^k = f_t^k(x_t^p) + F_t^k(x_t^p)x_t^k + G_t^k(x_t^p)u_t^k \quad (2.25b)$$

$$y_t = h_t(x_t^p) + H_t(x_t^p)x_t^k + e_t \quad (2.25c)$$

$$u_t = \begin{bmatrix} u_t^p \\ u_t^k \end{bmatrix} \sim \mathcal{N}(0, Q_t), Q_t = \begin{bmatrix} Q_t^p & M_t \\ M_t^T & Q_t^k \end{bmatrix}, Q_t^p > 0, \quad (2.25d)$$

$$x_0^k \sim \mathcal{N}(\hat{x}_0^k, P_0^k), P_0^k > 0, \quad (2.25e)$$

$$e_t \sim \mathcal{N}(0, R_t), R_t > 0, \quad (2.25f)$$

where $x_t = [x_t^p, x_t^k]^T$ and x_0^k , u_t and e_t are independent.

The aim is still to estimate the posterior distribution recursively

$$p(x_t|Y_t) = p(x_t^p, x_t^k|Y_t). \quad (2.26)$$

To use the structure of the problem we consider the posterior $p(X_t^p, x_t^k|Y_t)$. Bayes' rule states that

$$p(X_t^p, x_t^k|Y_t) = p(x_t^k|X_t^p, Y_t) p(X_t^p|Y_t). \quad (2.27)$$

We can use an extended Kalman filter to estimate $p(x_t^k|X_t^p, Y_t)$ and a particle filter to estimate $p(X_t^p|Y_t)$. Now we get better accuracy than a particle filter used for the entire state vector x_t , assuming the same number of samples.

In the model (2.25) there are two state transition equations. This means that by sampling new samples, x_{t+1}^p , for the particle filter, actually give information about the other part of the state vector, x_t^k . The equations for the Kalman filter, presented in Section 2.1, can be adjusted to take this information into consideration when estimating

x_t^k . During the time update there is a second measurement update, that comes from the new particle filter samples. This new measurement update is not quite the same as the first, since the process noise is correlated for x^p and x^k .

$$p(x_t^k | X_t^p, Y_t) = \mathcal{N}(\hat{x}_{t|t}^k, P_{t|t}^k) \quad (2.28a)$$

$$p(x_{t+1}^k | X_{t+1}^p, Y_t) = \mathcal{N}(\hat{x}_{t+1|t}^k, P_{t+1|t}^k), \quad (2.28b)$$

where

$$\begin{aligned} \hat{x}_{t|t}^k &= \hat{x}_{t|t-1}^k + K_{f,t} (y_t - h_t(x_t^p) - H_t \hat{x}_{t|t-1}^k) \\ P_{t|t}^k &= P_{t|t-1}^k - K_{f,t} S_{f,t} K_{f,t}^T \\ K_{f,t} &= P_{t|t-1}^k H_t^T S_{f,t}^{-1} \\ S_{f,t} &= R_t + H_t P_{t|t-1}^k H_t^T, \end{aligned} \quad (2.29)$$

and

$$\begin{aligned} \hat{x}_{t+1|t}^k &= (\bar{F}_t^k - K_{p,t} F_t^p) \hat{x}_{t|t}^k + f_t^p(x_t^p) \\ &\quad + (D_t + K_{p,t}) (x_{t+1}^p - f_t^p(x_t^p)) \\ P_{t+1|t}^k &= \bar{F}_t^k P_{t|t}^k \bar{F}_t^{kT} + G_t^k \bar{Q}_t^k G_t^{kT} - K_{p,t} S_{p,t} K_{p,t}^T \\ K_{p,t} &= \bar{F}_t^k P_{t|t}^k F_t^{pT} S_{p,t}^{-1} \\ S_{p,t} &= G_t^p Q_t^p G_t^{pT} + F_t^p P_{t|t}^k F_t^{pT}, \end{aligned} \quad (2.30)$$

with the following definitions

$$\begin{aligned} D_t &= G_t^k M_t^T (G_t^p Q_t^p)^{-1} \\ \bar{F}_t^k &= F_t^k - D_t F_t^p \\ \bar{Q}_t^k &= Q_t^k - M_t^T (Q_t^p)^{-1} M_t. \end{aligned} \quad (2.31)$$

By repeated use of the Bayes' rule we can write

$$p(X_t^p | Y_t) = \frac{p(y_t | X_t^p, Y_{t-1}) p(x_t^p | X_{t-1}^p, Y_{t-1})}{p(y_t | Y_{t-1})} p(X_{t-1}^p | Y_{t-1}). \quad (2.32)$$

In (2.32), the two probability densities in the numerator both have contributions from x_t^k according to

$$p(y_{t+1} | X_{t+1}^p, Y_t) = \mathcal{N}(h(x_{t+1}^p) + H_{t+1} \hat{x}_{t+1|t}^k, R_{t+1} + H_{t+1} P_{t+1|t}^k H_{t+1}^T) \quad (2.33a)$$

$$p(x_{t+1}^p | X_t^p, Y_t) = \mathcal{N}(f_t^p(x_t^p) + F_t^p \hat{x}_{t|t}^k, F_t^p P_{t|t}^k F_t^{pT} + G_t^p Q_t^p G_t^{pT}). \quad (2.33b)$$

For the particle filter we can chose

$$q(x_t^p | X_{t-1}^p, Y_t) = p(x_t^p | X_{t-1}^p, Y_{t-1}), \quad (2.34)$$

and the update equation for the importance weights becomes

$$w(X_t^p) = p(y_t | X_{t-1}^p, Y_{t-1}) w(X_{t-1}^p). \quad (2.35)$$

Algorithm 2.3 gives the details for the Rao-Blackwellized particle filter.

Algorithm 2.3 (The Rao-Blackwellized particle filter).

At time $t = 0$:

1. For $i = 1, \dots, N$, sample $x_0^{p,(i)} \sim p(x_0^p)$, and set $\{\hat{x}_{0|-1}^{k,(i)}, P_{0|-1}^{k,(i)}\} = \{0, P_0^k\}$.
2. For $i = 1, \dots, N$, compute $w_0^{(i)} = p(y_0 | x_0^{p,(i)})$ and normalize $\bar{w}_0^{(i)} = \frac{w_0^{(i)}}{\sum_j w_0^{(j)}}$.
3. For $i = 1, \dots, N$, compute $\{\hat{x}_{0|0}^{k,(i)}, P_{0|0}^{k,(i)}\}$.

For each time $t \geq 1$:

1. If resampling (e.g., $N_{\text{eff}} < N_{\text{th}}$) then resample and set the weights $\bar{w}_{t-1}^{(i)} = \frac{1}{N}$.
2. For $i = 1, \dots, N$, sample $x_t^{p,(i)} \sim p(x_t^p | X_{t-1}^{p,(i)}, Y_{t-1})$.
3. For $i = 1, \dots, N$, compute $\{\hat{x}_{t|t-1}^{k,(i)}, P_{t|t-1}^{k,(i)}\}$.
4. For $i = 1, \dots, N$, update the weights $w_t^{(i)} = p(y_t | X_t^{p,(i)}, Y_{t-1}) \bar{w}_{t-1}^{(i)}$ and normalize $\bar{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_j w_t^{(j)}}$.
5. For $i = 1, \dots, N$, compute $\{\hat{x}_{t|t}^{k,(i)}, P_{t|t}^{k,(i)}\}$.

As described in Algorithm 2.3, there will be one Kalman filter running for each sample. This implies that there will be lots of time spent on updating all the Kalman filters. However, in some cases the covariance will be the same for all the samples. It can be seen from Equations (2.29) and (2.30) that the covariance will be the same for all samples if F_t^p , F_t^k , G_t^p , G_t^k and H_t are constant for all $x_t^{p,(i)}$. For example, in this thesis we will use \hat{x}_t^p for all samples when calculating F_t^p , F_t^k , G_t^p , G_t^k and H_t .

The estimates for \hat{x}_t^p , \hat{x}_t^k , and P_t^p are straightforward

$$\hat{x}_t^p = \sum_{i=1}^N \bar{w}_t^{(i)} x_t^{p,(i)} \quad (2.36a)$$

$$\hat{x}_t^k = \sum_{i=1}^N \bar{w}_t^{(i)} \hat{x}_{t|t}^{k,(i)} \quad (2.36b)$$

$$\hat{P}_t^p = \sum_{i=1}^N \bar{w}_t^{(i)} \left(x_t^{p,(i)} - \hat{x}_t^p \right) \left(x_t^{p,(i)} - \hat{x}_t^p \right)^T. \quad (2.36c)$$

Eventhough we are most interested in the states, we would also like to have an estimate of the covariance for the linear part. It can be shown, see e.g., Nordlund [14], that the covariance can be estimated by

$$\hat{P}_t^k = \sum_{i=1}^N \bar{w}_t^{(i)} \left(P_{t|t}^{k,(i)} + \left(\hat{x}_{t|t}^{k,(i)} - \hat{x}_t^k \right) \left(\hat{x}_{t|t}^{k,(i)} - \hat{x}_t^k \right)^T \right), \quad (2.36d)$$

i.e., the weighted sum of each samples covariance and spread of the mean.

2.5 Cramer-Rao bounds

For a certain model there is a lower bound on the estimation error given by the Cramer-Rao bound. This is not a lower bound for the estimation problem in total but only for the estimation problem when using the current model. See e.g., Bergman [2] or Karlsson et al. [10]. This bound is based on the amount of information at each iteration, and is useful when using for example a particle filter, since it is not guaranteed to be optimal as the Kalman filter.

For every iteration there is a lower bound for the covariance according to

$$\left(P_t^{CR} \right)^2 = Q_t Z_t^{-1} R, \quad (2.37)$$

where P_t^{CR} is the Cramer-Rao lower bound for the covariance. Z_t is a measurement of the amount of information at the current state, x_t . This is calculated as a 2 by 2 matrix of differentials of the state to measurement function as

$$Z_t = E \left[\left[\begin{pmatrix} \frac{\partial h(x)}{\partial x_n} \\ \frac{\partial h(x)}{\partial x_e} \end{pmatrix} \left(\frac{\partial h(x)}{\partial x_n} \quad \frac{\partial h(x)}{\partial x_e} \right) \right] \right]_{x=x_t}. \quad (2.38)$$

In Chapter 5 there will be some tests where this lower bound is used. If the Cramer-Rao bound is larger than the current covariance estimate there is not enough information available to keep the samples as tight as they are and they will therefore spread. This fact will be used.

Chapter 3

Integrated aircraft navigation

In this chapter aircraft navigation and the systems used for integrated navigation will be described. A model for the aircraft dynamics will also be described. Throughout this chapter the World Geodetic System 84, defined 1984, will be used. This is a standard reference coordinate system for the earth. For more details about the WGS84 and the basic dynamic equations, see e.g., NIMA [11]. The dynamic equations of the aircraft are from Nordlund [14].

3.1 Aircraft navigation

First we need a coordinate system to represent the position and velocity of the aircraft. In most cases the WGS84, World Geodetic System 84, is used. In this system the earth is represented as a reference ellipsoid and the position is described by two angles and an altitude above the ellipsoid surface. The altitude is measured along the surface normal. The first angle, latitude, is the angle between this normal and the plane through the equator. The second angle, longitude, is the angle between the projection of the surface normal onto the equator plane and the zero meridian that goes through Greenwich in the UK, see Figure 3.1. There are also other geocentric coordinate systems and coordinate systems that are attached to the aircraft that we will use later.

Not only the position and the velocity are important, but also the attitude of the aircraft, which is the orientation of the aircraft relative to the surface of the earth. Attitude is described by three angles, namely heading, roll and pitch. Heading is the angle between the projection of the axis going through the aircraft from rear to nose, also called

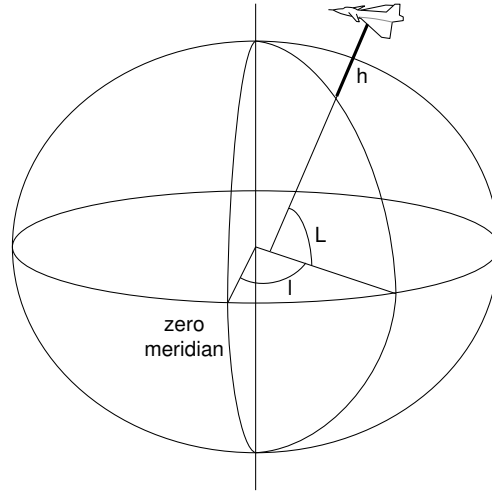


Figure 3.1: The WGS84 ellipsoid with the zero meridian, the two angles Latitude (L) and Longitude (l) and the altitude (h).

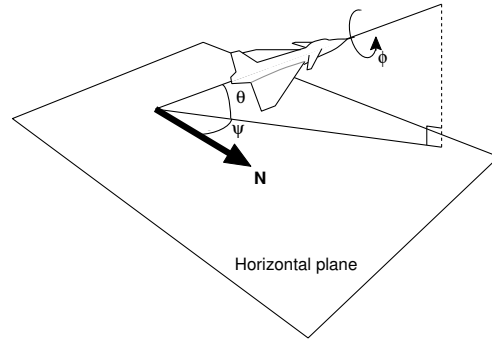


Figure 3.2: Heading, ψ , pitch, θ , and roll, ϕ , are the angles used to represent the orientation of the aircraft relative the horizontal plane.

the long axis, onto the horizontal plane below the aircraft and the direction to the north in the same plane. Pitch is the angle between the same projection of the long axis of the aircraft and the long axis itself. Roll is the angle between the axis going through the wing tips and the horizontal plane measured around the long axis. These three angles are notated ψ , θ and ϕ respectively. See Figure 3.2.

To be able to represent the dynamics of the aircraft position, veloc-

ity and attitude mathematically, we will need a number of coordinate frames given by:

Inertial frame (i): Often called the reference frame. This frame should be placed at some origin and neither accelerate nor rotate. It turns out that by placing this frame at the center of the earth, but not rotating with the earth, is good enough for this type of application. The z-axis points through the north pole.

Earth frame (e): This frame also has its origin at the center of the earth, but it also rotates with the earth. The position is often given relative to this frame as latitude, longitude and altitude. The z-axis points through the north pole of the earth, the x-axis through the zero-meridian and the y-axis completes the right hand system, i.e., points through the longitude EAST 90° .

Navigation frame (n): Here is the first of the two more application specific frames. The frame is placed at the center of the aircraft. The z-axis points downwards along the normal of the surface of the earth, the x-axis points to the north and the y-axis completes the right hand system, i.e., points to the east.

Body frame (b): The second application specific frame is the body frame. This frame is placed at the center of the aircraft, but it also rotates with the aircraft. The x-axis points through the nose of the aircraft, the z-axis down through the belly and the y-axis completes the right hand system, i.e., points through the right wing.

As stated above, WGS84 represents the earth by an ellipsoid that is circular along the equator and flattened at the poles. Table 3.1 lists some important parameters.

Table 3.1: WGS84 parameters

Parameter	Notation	Numerical value
Semimajor axis	r_0	$6.378137 \cdot 10^6 \text{m}$
Earth's ellipticity	e	$1/298.2573$
Angular velocity of the earth	ω_{ie}	$7.292115 \cdot 10^{-5} \text{rad/s}$
Gravity constant	g_0	9.805m/s^2
Empirical constant	J	0.00108
Earth's eccentricity	$\epsilon^2 = e(2 - e)$	

In Table 3.1, the *Semimajor axis* is the mean radius at the equator, *Earth ellipticity* and *eccentricity* are both numbers that state how flattened the ellipsoid is, *Angular velocity* is the rate of rotation of the

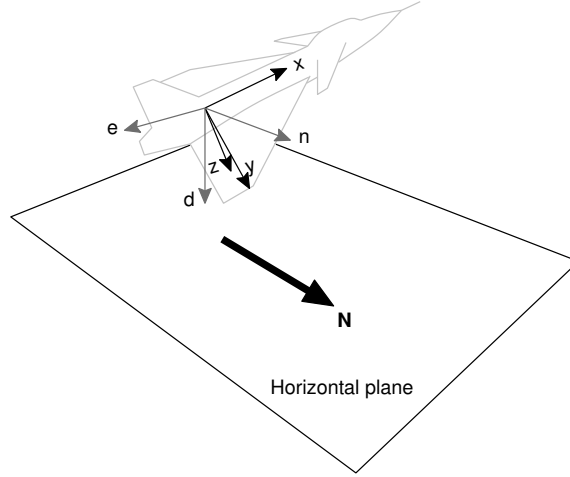


Figure 3.3: Naviation frame $\{n, e, d\}$ and body frame $\{x, y, z\}$.

earth around the z -axis of the inertial frame, and the *Gravity constant* and the *Empirical constant* are used when calculating the actual gravity force.

Now we have the coordinate frames and the reference ellipsoid and can write some basic dynamic equations. Latitude is denoted by L , longitude by l and altitude by h . The velocity relative to the earth frame expressed in the navigation frame is denoted by $v^n = [v_n \ v_e \ v_d]^T$ as a vector with its north, east and down components.

$$\dot{L} = \frac{v_n}{r_L + h} \quad (3.1a)$$

$$\dot{l} = \frac{v_e}{(r_l + h) \cos L} \quad (3.1b)$$

$$\dot{h} = -v_d, \quad (3.1c)$$

where the two radii of curvature are

$$r_L = \frac{r_0 (1 - \varepsilon^2)}{(1 - \varepsilon^2 \sin^2 L)^{\frac{3}{2}}} \quad (3.2a)$$

$$r_l = \frac{r_0}{(1 - \varepsilon^2 \sin^2 L)^{\frac{1}{2}}} \quad (3.2b)$$

The velocity is given as a solution to the differential equation

$$\dot{v}^n = C_b^n a^b - (\Omega_{en}^n + 2\Omega_{ie}^n) v^n. \quad (3.3)$$

Here C_b^n is the transformation matrix that transforms from the body frame to the navigation frame, this is a standard notation in this thesis. Ω is the skew-symmetric form

$$\Omega = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}$$

of the corresponding vector $\omega = [\omega_x \ \omega_y \ \omega_z]^T$ and are used to replace a cross product with a matrix multiplication. a^b is the specific acceleration vector in the body frame.

In vector form, the rotation of the earth frame relative the inertial frame and the rotation of the navigation frame relative the earth frame are

$$\omega_{ie}^n = \omega_{ie} \begin{bmatrix} \cos L \\ 0 \\ -\sin L \end{bmatrix}, \quad \omega_{en}^n = \begin{bmatrix} \dot{L} \cos L \\ -\dot{L} \\ -\dot{L} \sin L \end{bmatrix} = \begin{bmatrix} \frac{v_e}{r_l+h} \\ -\frac{v_n}{r_l+h} \\ -\frac{v_e \tan L}{r_l+h} \end{bmatrix}. \quad (3.4)$$

Moreover, $\omega_{in}^n = \omega_{ie}^n + \omega_{en}^n$.

The attitude is denoted by ϕ , θ and ψ for roll, pitch and heading respectively. The corresponding transformation matrix from body frame to navigation frame is

$$C_b^n = \begin{bmatrix} \cos \psi & -\sin \psi & 0 \\ \sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \quad (3.5a)$$

$$\dot{C}_b^n = C_b^n \Omega_{ib}^b - \Omega_{in}^n C_b^n, \quad (3.5b)$$

where ω_{ib}^b is the angular velocity of the body frame relative the inertial frame and can be measured with gyros.

These are some basic dynamic equations that are of importance in aircraft navigation and they will be used later in this chapter.

3.2 Inertial navigation

The idea of dead-reckoning is to calculate the current position from a known starting position by integrating the velocity. In an inertial navigation system, INS, the velocity is calculated by integrating the accelerations. The reason for this is that no external sensors are needed to get the acceleration. Not only the position and velocity are important to know in aircraft navigation, but also the attitude of the aircraft. This is the orientation of the aircraft relative to the surface of the earth and can be described by heading, pitch and roll. These are three angles indicating how much the aircraft is tilted along different

axis. INS therefore includes both accelerometers and angular velocity sensors. These sensors do not need any data from any other sensors to produce their data. This means that the INS is an independent sensor and is therefore often the central component of a modern integrated navigation system. The data from the sensors are integrated to calculate the velocity, position and attitude of the aircraft. Biases and noise in the sensors, and the fact that the calculations are performed in discrete timesteps, will make the data produced drift from the real position, velocity and attitude. Therefore the INS is often supported by observations made from other sensors. It turns out that it is enough to get observations of the position to support all output data from INS. One sensor that is often used is the GPS. The sensor fusion of INS and GPS can be linearized and a Kalman filter can be used, see Figure 3.4. However, GPS is a sensor that is dependent on data from outside the aircraft.

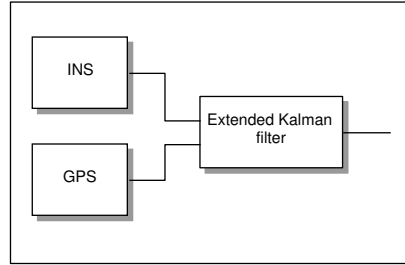


Figure 3.4: Sensor fusion of INS and GPS using an extended Kalman filter.

3.3 Terrain aided positioning

The idea is to use the information in the terrain below the aircraft to support the INS. A radar altimeter, measuring the ground clearance, along with the altitude of the aircraft is used to get the elevation of the ground below the aircraft. This elevation profile can be compared to a database to find the position of the aircraft. A particle filter or a point-mass filter is often used in the TAP. This position information is used in the same way as the GPS position, see Figure 3.5.

Terrain Aided Positioning (TAP) uses the height database, which is highly non-linear. The data from TAP is used by the sensorfusion to estimate the error in the INS. Using a linear filter means that we consider the data as the correct data with additional measurement noise. The data from TAP is more complicated than that, why a linear filter

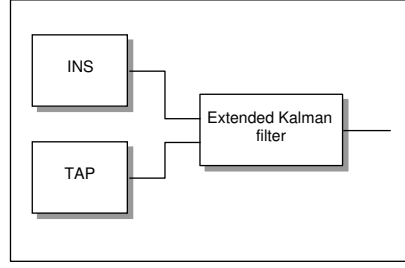


Figure 3.5: Sensor fusion of INS and TAP using an extended Kalman filter.

is not the best idea. Instead we will consider the radar altimeter as our supporting sensor and have the database as a part of the model. By doing this, the non-linearity in the database is a part of the model and we will have to use a non-linear filter, see Figure 3.6.

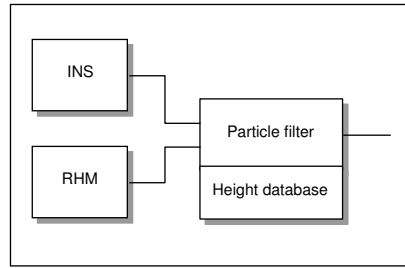


Figure 3.6: When using the radar altimeter (RHM) the height database will be a part of the model and a non-linear filter is therefore required.

3.4 Error dynamic model

The dynamics of an aircraft can be fairly fast. Consider for example the roll that can have angular rates up to $300^\circ/\text{s}$. Instead of focusing on the real states, the sensor fusion algorithm will try to estimate the state errors of the INS. These have significantly slower dynamics and are therefore much easier to estimate. Let x^{INS} denote the states from the INS, x be the real states and \tilde{x} be the error in the states, i.e.,

$$x = x^{INS} + \tilde{x}. \quad (3.6)$$

The sensor fusion will be slightly different than in Figure 3.6, see Figure 3.7.

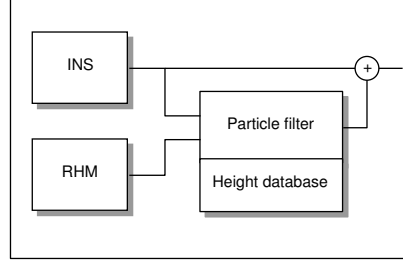


Figure 3.7: The algorithm estimates the error in the INS states instead of estimating the states directly. This will give better accuracy, due to the much slower dynamics of the errors.

This section will first investigate the dynamics of the aircraft and then derive the dynamics of the INS state errors. The result will be a model that can be used in the sensor fusion, described in Chapter 4. These error dynamic equations are typically non-linear, but they can be linearized without introducing any significant errors. The terrain height database can not be linearized and a non-linear estimation method will therefore be implemented. The terrain height database is only dependent on the horizontal position. If the error dynamic equations are linearized completely, the problem will satisfy the requirements for the Rao-Blackwellized filter. Also the horizontal position will be linearized in the error dynamic equations. From Section 3.1 we have some equations of the form

$$\dot{x} = f(x, u), \quad (3.7)$$

but we want the dynamic equations for the errors. By combining Equation (3.6) and (3.7), we can write the dynamics for the errors according to

$$\begin{aligned} \dot{\tilde{x}} &= \dot{x} - \dot{x}^{INS} = f(x, u) - f(x^{INS}, u^{INS}) \\ &= f(x, u) - f(x - \tilde{x}, u - \tilde{u}), \end{aligned} \quad (3.8)$$

assuming the dynamics for the INS states are the same as for the real states. \tilde{u} denotes the sensor errors. The dynamics in (3.8) are much slower than the dynamics in the real states, assuming that the sensor errors are small enough.

3.4.1 Position error dynamics

Combining Equation (3.8) with the dynamic expressions for L , l and h yields

$$\dot{\tilde{L}} = \frac{v_n}{r_L + h} - \frac{v_n - \tilde{v}_n}{r_L - \tilde{r}_L + h - \tilde{h}} \quad (3.9a)$$

$$\dot{\tilde{l}} = \frac{v_e}{(r_l + h) \cos L} - \frac{v_e - \tilde{v}_e}{(r_l - \tilde{r}_l + h - \tilde{h}) \cos(L - \tilde{L})} \quad (3.9b)$$

$$\dot{\tilde{h}} = -v_d - (-(v_d - \tilde{v}_d)) = -\tilde{v}_d. \quad (3.9c)$$

\tilde{v}_n and \tilde{v}_e already enters these equations linearly. \tilde{h} , on the other hand, do have to be linearized and also \tilde{L} will be linearized. This will not introduce any significant error. Using Equation (3.2) and Taylor expansion around h and ε^2 yields

$$\frac{1}{r_L + h} = \frac{1}{r_0} - \frac{h}{r_0^2} + \frac{2\varepsilon^2 - 3\varepsilon^2 \sin^2 L}{2r_0} + \mathcal{O}\left(\frac{\varepsilon^4}{r_0}\right) \quad (3.10a)$$

$$\frac{1}{r_l + h} = \frac{1}{r_0} - \frac{h}{r_0^2} - \frac{\varepsilon^2 \sin^2 L}{2r_0} + \mathcal{O}\left(\frac{\varepsilon^4}{r_0}\right). \quad (3.10b)$$

Using the Taylor expansion $\sin^2(L + \tilde{L}) = \sin^2 L + \mathcal{O}(\tilde{L})$ gives

$$\frac{1}{r_L - \tilde{r}_L + h - \tilde{h}} = \frac{1}{r_0} - \frac{h}{r_0^2} + \frac{2\varepsilon^2 - 3\varepsilon^2 \sin^2 L}{2r_0} + \mathcal{O}\left(\frac{\varepsilon^4}{r_0}\right) + \mathcal{O}\left(\frac{\tilde{h}}{r_0^2}\right) \quad (3.11a)$$

$$\frac{1}{r_l - \tilde{r}_l + h - \tilde{h}} = \frac{1}{r_0} - \frac{h}{r_0^2} - \frac{\varepsilon^2 \sin^2 L}{2r_0} + \mathcal{O}\left(\frac{\varepsilon^4}{r_0}\right) + \mathcal{O}\left(\frac{\tilde{h}}{r_0^2}\right). \quad (3.11b)$$

Finally, using the Taylor expansion

$$\frac{1}{\cos(L - \tilde{L})} = \frac{1}{\cos L} + \frac{\sin L}{\cos^2 L} \tilde{L} + \mathcal{O}\left(\frac{\tilde{L}^2}{\cos^3 L}\right),$$

we get

$$\dot{\tilde{L}} = \frac{\tilde{v}_n}{r_0} + \mathcal{O}\left(\frac{\tilde{v}_n \varepsilon^2}{r_0}\right) + \mathcal{O}\left(\frac{v_n \tilde{h}}{r_0^2}\right) \quad (3.12a)$$

$$\dot{\tilde{l}} = \frac{v_e \tilde{L} \sin L}{r_0 \cos^2 L} + \frac{\tilde{v}_e}{r_0 \cos L} + \mathcal{O}\left(\frac{\tilde{v}_e \varepsilon^2}{r_0 \cos L}\right) + \mathcal{O}\left(\frac{v_e \tilde{h}}{r_0^2 \cos L}\right). \quad (3.12b)$$

These are the linearized dynamic equations for the latitude and longitude errors. It can be verified that the \mathcal{O} -terms in Equation (3.12)

are negligible for the values involved. From Equation (3.9) we see that the error equation for the altitude error, \tilde{h} , is simply

$$\dot{\tilde{h}} = -\tilde{v}_d. \quad (3.13)$$

The altitude state turns out to be a little special. We will get back to this in Section 3.4.3.

3.4.2 Velocity error dynamics

In INS, the accelerometers measure the specific acceleration, $f^{b,INS}$, in a way so that they include the effect of the gravity, even though the aircraft does not accelerate accordingly. This means that it might seem like the aircraft is accelerating by g upwards. The relationship between the real specific acceleration and the measured is given by

$$f^{b,INS} = a^b - g^b - \tilde{a}^b = f^b - \tilde{a}^b, \quad (3.14)$$

where a^b is the actual acceleration, g^b is the gravity acceleration and \tilde{a}^b is the sensor error. The gravity vector g^n is really a sum of the earth's gravitation, G^n , and the centripetal acceleration due to the rotation of the earth

$$g^n = G^n - \Omega_{ie}^n \Omega_{ie}^n r^n, \quad (3.15)$$

where r^n is the position vector of the aircraft measured from the center of the earth given in the navigation frame. Since the reference ellipsoid is almost spherical and the centripetal acceleration is small compared to the earth's gravity, we can approximate the gravity by its z -component in the navigation frame

$$g^n \approx [0 \ 0 \ g_d]^T \quad (3.16)$$

$$g_d = g_0 \left(1 - 2 \frac{h}{r_0} + \varepsilon^2 \sin^2 L \right) + \frac{2}{3} J g_0 (1 - 3 \sin^2 L) \quad (3.17)$$

$$- r_0 \omega_{ie}^2 \left(1 - \frac{\varepsilon^2}{2} \sin^2 L + \frac{h}{r_0} \right) (1 - \sin^2 L). \quad (3.18)$$

Combining Equations (3.3), (3.8) and (3.14) gives

$$\begin{aligned} \dot{\tilde{v}}^n &= C_b^n f^b + g^n - (\Omega_{en}^n + 2\Omega_{ie}^n) v^n - \left((C_b^n - \tilde{C}_b^n) (f^b - \tilde{a}^b) + \right. \\ &\quad \left. g^n - \tilde{g}^n - \left(\Omega_{en}^n - \tilde{\Omega}_{en}^n + 2 \left(\Omega_{ie}^n - \tilde{\Omega}_{ie}^n \right) \right) (v^n - \tilde{v}^n) \right) = \\ &= \tilde{C}_b^n f^b + \left(C_b^n - \tilde{C}_b^n \right) \tilde{a}^b + \tilde{g}^n - \left(\tilde{\Omega}_{en}^n + 2\tilde{\Omega}_{ie}^n \right) v^n - \\ &\quad \left(\Omega_{en}^n - \tilde{\Omega}_{en}^n + 2 \left(\Omega_{ie}^n - \tilde{\Omega}_{ie}^n \right) \right) \tilde{v}^n, \end{aligned} \quad (3.19)$$

where the corresponding angular velocity errors, $\tilde{\Omega}_{en}^n$ and $\tilde{\Omega}_{ie}^n$, are,

$$\tilde{\omega}_{en}^n = \begin{bmatrix} \frac{v_e}{r_l+h} \\ -\frac{v_n}{r_L+h} \\ -\frac{v_e \tan L}{r_l+h} \end{bmatrix} - \begin{bmatrix} \frac{v_e - \tilde{v}_e}{r_l - \tilde{r}_l + h - \tilde{h}} \\ -\frac{v_n - \tilde{v}_n}{r_L - \tilde{r}_L + h - \tilde{h}} \\ -\frac{(v_e - \tilde{v}_e) \tan(L - \tilde{L})}{r_l - \tilde{r}_l + h - \tilde{h}} \end{bmatrix} \quad (3.20a)$$

$$\tilde{\omega}_{ie}^n = \omega_{ie} \begin{bmatrix} \cos L \\ 0 \\ -\sin L \end{bmatrix} - \omega_{ie} \begin{bmatrix} \cos(L - \tilde{L}) \\ 0 \\ -\sin(L - \tilde{L}) \end{bmatrix}. \quad (3.20b)$$

Let us consider the attitude error for a while. $C_n^{n,INS}$ denotes the transformation matrix from the navigation frame to the navigation frame of the INS attitude. Assuming that the attitude error is small, we can define a small angle transformation $\gamma^n = [\gamma_n \ \gamma_e \ \gamma_d]^T$ in skew-symmetric matrix form denoted by Γ^n , through

$$\tilde{C}_b^n = C_b^n - C_b^{n,INS} = (I - C_n^{n,INS}) C_b^n = \Gamma^n C_b^n + \mathcal{O}((\Gamma^n)^2). \quad (3.21)$$

The three angles in γ^n do not correspond directly to heading, roll and pitch. The reason for introducing them is that it is an easy way to limit the number of states that needs to be estimated, in this case the three angles in γ^n instead of six in a more general transformation matrix. The limitation is that γ^n is only valid for small angles. By using the earlier linearizations and the Taylor expansions ($|L| < \pi/2$)

$$\tan(L - \tilde{L}) = \tan L - \frac{\tilde{L}}{\cos^2 L} + \mathcal{O}(\tilde{L}^2) \quad (3.22a)$$

$$\sin(L - \tilde{L}) = \sin L - \tilde{L} \cos L + \mathcal{O}(\tilde{L}^2) \quad (3.22b)$$

$$\cos(L - \tilde{L}) = \cos L + \tilde{L} \sin L + \mathcal{O}(\tilde{L}^2), \quad (3.22c)$$

Equation (3.20) can be linearized according to

$$\tilde{\omega}_{en}^n = \underbrace{\begin{bmatrix} \frac{\tilde{v}_e}{r_0} \\ -\frac{\tilde{v}_n}{r_0} \\ -\frac{v_e \tilde{L}}{r_0 \cos^2 L} - \frac{\tilde{v}_e \tan L}{r_0} \end{bmatrix}}_{\hat{\omega}_{en}^n} + \begin{bmatrix} \mathcal{O}\left(\frac{\tilde{v}_e \varepsilon^2}{r_0}\right) + \mathcal{O}\left(\frac{v_e \tilde{h}}{r_0^2}\right) \\ \mathcal{O}\left(\frac{\tilde{v}_n \varepsilon^2}{r_0}\right) + \mathcal{O}\left(\frac{v_n \tilde{h}}{r_0^2}\right) \\ \mathcal{O}\left(\frac{\tilde{v}_e \varepsilon^2 \tan L}{r_0}\right) + \mathcal{O}\left(\frac{v_e \tilde{h} \tan L}{r_0^2}\right) \end{bmatrix} \quad (3.23a)$$

$$\tilde{\omega}_{ie}^n = \underbrace{\omega_{ie} \begin{bmatrix} -\tilde{L} \sin L \\ 0 \\ -\tilde{L} \cos L \end{bmatrix}}_{\hat{\omega}_{ie}^n} + \omega_{ie} \begin{bmatrix} \mathcal{O}(\tilde{L}^2) \\ 0 \\ \mathcal{O}(\tilde{L}^2) \end{bmatrix}. \quad (3.23b)$$

These linearizations along with the simplifications

$$\omega_{en}^n - \tilde{\omega}_{en}^n = \underbrace{\begin{bmatrix} \frac{v_e}{r_0} \\ -\frac{v_n}{r_0} \\ -\frac{v_e \tan L}{r_0} \end{bmatrix}}_{\tilde{\omega}_{en}^n} + \begin{bmatrix} \mathcal{O}\left(\frac{\tilde{v}_e}{r_0}\right) + \mathcal{O}\left(\frac{\varepsilon^2 v_e L}{r_0}\right) \\ \mathcal{O}\left(\frac{\tilde{v}_n}{r_0}\right) + \mathcal{O}\left(\frac{\varepsilon^2 v_n L}{r_0}\right) \\ \mathcal{O}\left(\frac{\tilde{v}_e \tan L}{r_0}\right) + \mathcal{O}\left(\frac{\varepsilon^2 v_e L \tan L}{r_0}\right) \end{bmatrix} \quad (3.24a)$$

$$\omega_{ie}^n - \tilde{\omega}_{ie}^n = \underbrace{\omega_{ie} \begin{bmatrix} \cos L \\ 0 \\ -\sin L \end{bmatrix}}_{\omega_{ie}^n} + \omega_{ie} \begin{bmatrix} \mathcal{O}(\tilde{L}) \\ 0 \\ \mathcal{O}(\tilde{L}) \end{bmatrix}, \quad (3.24b)$$

give us the following linearized expression for the velocity error

$$\dot{\tilde{v}}^n = \Gamma^n \underbrace{C_b^n f^b}_{f^n} + C_b^n \tilde{a}^b - \left(\hat{\tilde{\Omega}}_{en}^n + 2\hat{\tilde{\Omega}}_{ie}^n \right) v^n - \left(\hat{\tilde{\Omega}}_{en}^n + 2\hat{\tilde{\Omega}}_{ie}^n \right) \tilde{v}^n + \tilde{g}^n. \quad (3.25)$$

3.4.3 Altitude error dynamics

As stated in Section 3.4.2, the error dynamics for the altitude is special for the reason that it is unstable. Let us start by differentiating Equation (3.13)

$$\ddot{\tilde{h}} = -\dot{\tilde{v}}_d. \quad (3.26)$$

From Equation (3.25) it can be shown that the most significant term feeding $\dot{\tilde{v}}_d$ is the gravity error caused by the altitude error, i.e.,

$$\ddot{\tilde{h}} \approx \frac{2g_0}{r_0} \tilde{h}. \quad (3.27)$$

This second order differential equation has the solution

$$\tilde{h}(t) \approx \frac{\tilde{h}(0)}{2} \left(e^{\sqrt{\frac{2g_0}{r_0}} t} + e^{-\sqrt{\frac{2g_0}{r_0}} t} \right), \quad (3.28)$$

where \tilde{h}_t is unstable with the time constant $\sqrt{\frac{r_0}{2g_0}} \approx 10$ minutes. This means that the altitude channel have to be stabilized. One way to stabilize it is to use the altitude estimate from TAP. To do that we will have to have TAP running beside this new sensor fusion. Instead the altitude is stabilized by an altitude provided by a pressure sensor. This means that the altitude error does not follow the equations described above. Also, the velocity's down component, v_d , will be stabilized in

the process. Since the contribution of v_d to the total velocity is rather small, we will ignore the error in v_d . Instead the altitude error will be estimated by a random walk process

$$\dot{\tilde{h}} \approx u^h, \quad (3.29)$$

where u^h is independent Gaussian noise.

3.4.4 Attitude error dynamics

Only the attitude error dynamics remains to investigate. It turns out to be quite complicated equations involved, hence only the result will be presented here. For example in Nordlund [14], there is a derivation of this result

$$\dot{\gamma}^n = C_b^n \tilde{\omega}_{ib}^b - \tilde{\omega}_{in}^n - \Omega_{in}^n \gamma^n + [1 \quad 1 \quad 1]^T \mathcal{O} \left(\left\| (\Gamma^n)^2 \Omega_{ib}^b \right\|_\infty \right), \quad (3.30)$$

where $\tilde{\omega}_{ib}^b$ is the sensor noise in the angular velocity sensors in the INS.

3.4.5 Sensor noise

Now we have linear error dynamic equations for all the states we are interested in. When it comes to the sensors, they might be rather complicated. The easiest way to model them is by white Gaussian noise. Unfortunately this is not good enough for the accelerometers. They also have a bias that we need to consider. The bias in the altitude channel is already stabilized, and we will only consider the other two. Let us call the biases b_x^a and b_y^a for the accelerometer bias in x-direction and y-direction respectively and model them with a first order Gauss-Markov processes

$$\dot{b}^a = -\frac{1}{\tau} b^a + u^b \approx u^b. \quad (3.31)$$

Here, the last approximation is motivated by a very large time constant. As for the angular velocity sensors, it is close enough to model their errors by white noise.

3.4.6 Continuous state space equations

In the preceding sections the linearized dynamic equations for the INS errors have been derived. This section will summarize by defining the state vector to be used in the sensor fusion and also define the process noise vector. The state vector does not only contain the INS errors but also the biases of the accelerometers

$$x = \begin{bmatrix} \tilde{L} & \tilde{l} & \tilde{h} & \tilde{v}_n & \tilde{v}_e & \gamma_n & \gamma_e & \gamma_d & b_x^a & b_y^a \end{bmatrix}^T. \quad (3.32)$$

In Equation (3.25) the sensor noise is \tilde{a}^b and corresponds to the noise in the accelerometers. This is a noise on top of the bias that was discussed in Section 3.4.5. The noise in the altitude channel is straightforward from Equation (3.29), as is the noise of the Gauss-Markov process for the accelerometer biases in Equation (3.31). From Equation (3.30), the sensor noise from the angular velocity sensors is in $\tilde{\omega}_{ib}^b$. These will be the noise vector u

$$u = [u^h \quad u_x^a \quad u_y^a \quad u_x^\gamma \quad u_y^\gamma \quad u_z^\gamma \quad u_x^b \quad u_y^b]^T. \quad (3.33)$$

Combining Equations (3.12), (3.29), (3.25), (3.30) and (3.31) yield the state space equation

$$\dot{x}(t) = A(t)x(t) + B(t)u(t). \quad (3.34)$$

The expressions for $A(t)$ and $B(t)$ can be found in Appendix A.

3.4.7 Discrete state space equations

In this application the discrete version of the state space equations are used. Standard discretization techniques give

$$x_{t+1} = F_t x_t + G_t u_t, \quad (3.35)$$

where

$$F_t = I + T_s A_t \quad (3.36a)$$

$$G_t = T_s \left(I + \frac{T_s}{2} A_t \right) B_t, \quad (3.36b)$$

and T_s is the sampling period. For more details about this discretization see e.g., Rugh [15] or Jazwinski [9].

3.4.8 Summary

These discrete state space equations are the ones that will be used in this thesis. In Chapter 4, a Rao-Blackwellized particle filter will be created for these equations. By this we will have a running algorithm for the sensor fusion of the INS and the radar altimeter.

Chapter 4

Sensor fusion

In Chapter 3 the discrete state space equations for the filtering problem were derived. In this chapter the problem will be split into three parts and the different filtering methods will be adapted accordingly. There is a highly non-linear height database involved and a Rao-Blackwellized filter will therefore be used. The measurement noise from the radar altimeter is not pure Gaussian, but a Gaussian mixture. This is described in more detail in Section 4.3.

4.1 Problem decomposition

The measurements from the radar altimeter will be used to estimate the ground elevation. The difference between the altitude from the INS, h_t^{INS} , and the ground clearance from the radar altimeter, h_t^{rhm} , will be used as the observations, y_t , of the ground elevation.

$$\begin{aligned} y_t &= h_t^{INS} - h_t^{rhm} = h_t - h_t^{rhm} - \tilde{h}_t \\ &= h \left(\begin{bmatrix} L_t \\ l_t \end{bmatrix} + x_t^p \right) - x_t^g + e_t, \end{aligned} \quad (4.1)$$

where $x_t^p = \begin{bmatrix} \tilde{L}_t & \tilde{l}_t \end{bmatrix}^T$, $x_t^g = \tilde{h}_t$, e_t is the measurement noise and $h(\cdot)$ is the height database. x_t^p has its name from the fact that it will be estimated by a particle filter. The measurement noise is from the radar altimeter and is unfortunately not pure Gaussian, but a Gaussian mixture. Since the altitude error is easily separated from the rest of the problem it will be estimated by a multiple model method, e.g., GPB1,

hence the name x_t^g . Let

$$x_t^k = [\tilde{v}_{n,t} \quad \tilde{v}_{e,t} \quad \gamma_{n,t} \quad \gamma_{e,t} \quad \gamma_{d,t} \quad b_{x,t}^a \quad b_{y,t}^a]^T \quad (4.2a)$$

$$u_t^g = u_t^h \quad (4.2b)$$

$$u_t^k = [u_x^a \quad u_y^a \quad u_x^\gamma \quad u_y^\gamma \quad u_z^\gamma \quad u_x^b \quad u_y^b]^T \quad (4.2c)$$

$$F_t = \begin{bmatrix} F_{p,t}^p & 0 & F_{k,t}^p \\ 0 & 1 & 0 \\ F_{p,t}^k & 0 & F_{k,t}^k \end{bmatrix}, G_t = \begin{bmatrix} 0 & G_t^p \\ G_t^g & 0 \\ 0 & G_t^k \end{bmatrix} \quad (4.2d)$$

and rewrite the state space equation according to

$$x_{t+1}^p = F_{p,t}^p x_t^p + F_{k,t}^p x_t^k + \underbrace{G_t^p u_t^k + u_t^{add}}_{u_t^p} \quad (4.3a)$$

$$x_{t+1}^g = x_t^g + G_t^g u_t^g \quad (4.3b)$$

$$x_{t+1}^k = F_{p,t}^k x_t^p + F_{k,t}^k x_t^k + G_t^k u_t^k \quad (4.3c)$$

$$y_t = h \left(\begin{bmatrix} L_t^{INS} \\ l_t^{INS} \end{bmatrix} + x_t^p \right) - x_t^g + e_t \quad (4.3d)$$

where u_t^{add} is added noise to help the particle filter to compensate for the discretization. In this thesis u_t^{add} is assumed to be much bigger than $G_t^p u_t^k$ which means that the process noise for x^p and the process noise for x^k are independent. This added noise, also called roughening noise, was first proposed by Gordon [7].

4.2 Horizontal position error

The horizontal position error will be estimated by a particle filter according to the model

$$x_{t+1}^p = F_{p,t}^p x_t^p + F_{k,t}^p x_t^k + u_t^p \quad (4.4a)$$

$$y_t = h \left(\begin{bmatrix} L_t^{INS} \\ l_t^{INS} \end{bmatrix} + x_t^p \right) - x_t^g + e_t \quad (4.4b)$$

where $u_t^p \sim \mathcal{N}(0, Q^{add})$.

4.3 Altitude error

When measuring the ground clearance, the radar altimeter will sometimes react on echoes from tree tops. There is another database that describes what kind of terrain the aircraft is passing. This can be used to estimate the behavior of the radar altimeter noise. In this thesis

we will use one model for the noise no matter what kind of terrain is passed. The noise is modelled with two Gaussians, see Figure 4.1. The reason for this will be discussed in Section 5.2.4.

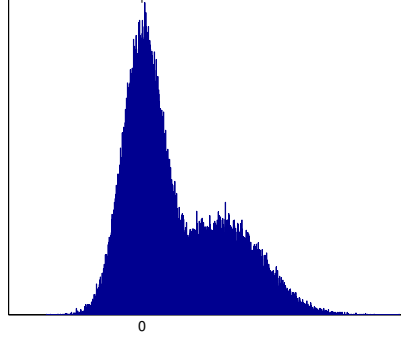


Figure 4.1: Figure of the radar altimeter noise.

This can be seen as a Gaussian mixture and will therefore be estimated with a multiple model method. The state space model for the altitude error is

$$x_{t+1}^g = x_t^g + G_t^g u_t^g \quad (4.5a)$$

$$y_t^g = y_t - h(\dots + x_t^p) = -x_t^g + e_t(\lambda_t) \quad (4.5b)$$

$$e_t(\lambda_t) \sim \begin{cases} \mathcal{N}(m_1, \sigma_1 + R), & \lambda_t = 1 \\ \mathcal{N}(m_2, \sigma_2 + R), & \lambda_t = 2 \end{cases} \quad (4.5c)$$

where h is the height database. There are some important parameters needed for the multiple model filter. First m_1 , m_2 , σ_1 and σ_2 correspond to the two cases when the radar echo is from the ground and from the tree tops. R is the uncertainty in the map and is just added to the covariance of the measurement noise. Last the mode transition matrix defined by $\pi_{kl} = p(\lambda_t = k | \lambda_{t-1} = l)$ is

$$\Pi = \begin{bmatrix} \pi_{11} & \pi_{12} \\ \pi_{21} & \pi_{22} \end{bmatrix} = \begin{bmatrix} \pi_1 & \pi_1 \\ \pi_2 & \pi_2 \end{bmatrix} \quad (4.6)$$

i.e., it is assumed that λ_t is independent of the mode history. The algorithm from Section 2.2 will be used.

4.4 Conditionally linear part

Only x_t^k is left to estimate. When considering x_t^p known there is no coupling from the measurements y_t to x_t^k . Instead we can use the time

update equation for x_t^p as measurement equation, since the samples in x_t^p evolves stochastically and the process noise for x_t^p is independent from the process noise for x_t^k . The state space model for x_t^k becomes

$$x_{t+1}^k = F_{p,t}^k x_t^p + F_{k,t}^k x_t^k + G_t^k u_t^k \quad (4.7a)$$

$$y_t^k = x_{t+1}^p - F_{p,t}^p x_t^p = F_{k,t}^p x_t^k + u_t^p \quad (4.7b)$$

and will be estimated by a Kalman filter. To calculate the observations we will need x_{t+1}^p and according to Section 4.2 we need x_t^k to sample x_{t+1}^p . Since both of them need each other, we will use $\hat{x}_{t|t-1}^k$ from the Kalman filter to sample x_{t+1}^p . To keep the computations down, the same $F_{p,t}^p$, $F_{k,t}^p$, $F_{p,t}^k$, $F_{k,t}^k$ and G_t^k will be used for all samples. This implies that the covariance estimate in the Kalman filter is the same for all samples and will only be calculated ones per iteration.

4.5 Sensor fusion

Here is the algorithm for the sensor fusion as described in the three preceding sections. The different simulations made and the results obtained are described in Chapter 5.

To initialize the particle filter, the samples, $x_0^{p,(i)}$, are sampled from the prior distribution and the weights are all set to $\frac{1}{N}$. The altitude and the linear states are all initialized to 0 and their covariances are set to their initial values. These are presented in Chapter 5. Each iteration is split into four parts. First is the GPB filter, then comes the particle filter. It turns out that the weights from the GPB filter can be used to calculate the particle filter weights, why the GPB filter is first. In the sample propagation step in the particle filter $\hat{x}_{t|t-1}^{k,(i)}$ is used since $\hat{x}_{t|t}^{k,(i)}$ has not been calculated yet. The reason for this is that the next set of samples, $x_{t+1}^{p,(i)}$, are needed to update the linear states. The last filter is the Kalman filter and after that the estimates are calculated as the appropriate weighted sums of the samples.

Algorithm 4.1 (The sensor fusion).

At time $t = 0$:

1. For $i = 1, \dots, N$, sample $x_0^{p,(i)} \sim p(x_0)$
and set $\bar{w} = \frac{1}{N}$
2. For $i = 1, \dots, N$, set $\hat{x}_{0|-1}^{g,(i)} = 0$ and $P_{0|-1}^{g,(i)} = P_0^g$
3. For $i = 1, \dots, N$, set $\hat{x}_{0|-1}^{k,(i)} = 0$ and $P_{0|-1}^k = P_0^k$

For each time $t \geq 0$:

GPB1

1. Measurement update and split

$$\begin{aligned}
 \hat{x}_{t|t}^{g1,(i)} &= \hat{x}_{t|t-1}^{g,(i)} - P_{t|t-1}^{g,(i)} \left(S_t^{g1,(i)} \right)^{-1} \left(y_t^{k,(i)} - m_1 + \hat{x}_{t|t-1}^{g,(i)} \right) \\
 \hat{x}_{t|t}^{g2,(i)} &= \hat{x}_{t|t-1}^{g,(i)} - P_{t|t-1}^{g,(i)} \left(S_t^{g2,(i)} \right)^{-1} \left(y_t^{k,(i)} - m_2 + \hat{x}_{t|t-1}^{g,(i)} \right) \\
 P_{t|t}^{g1,(i)} &= P_{t|t-1}^{g,(i)} - P_{t|t-1}^{g,(i)} \left(S_t^{g1,(i)} \right)^{-1} P_{t|t-1}^{g,(i)T} \\
 P_{t|t}^{g2,(i)} &= P_{t|t-1}^{g,(i)} - P_{t|t-1}^{g,(i)} \left(S_t^{g2,(i)} \right)^{-1} P_{t|t-1}^{g,(i)T} \\
 S_t^{g1,(i)} &= R + \sigma_1 + P_{t|t-1}^{g,(i)} \\
 S_t^{g2,(i)} &= R + \sigma_2 + P_{t|t-1}^{g,(i)}
 \end{aligned}$$

2. Weights

$$\begin{aligned}
 \alpha_t^{g1,(i)} &= \pi_1 \gamma \left(y_t^{k,(i)} - m_1 + \hat{x}_{t|t-1}^{g,(i)}, S_t^{g1,(i)} \right) \\
 \alpha_t^{g2,(i)} &= \pi_2 \gamma \left(y_t^{k,(i)} - m_2 + \hat{x}_{t|t-1}^{g,(i)}, S_t^{g2,(i)} \right) \\
 \bar{\alpha}_t^{g1,(i)} &= \frac{\alpha_t^{g1,(i)}}{\alpha_t^{g1,(i)} + \alpha_t^{g2,(i)}} \\
 \bar{\alpha}_t^{g2,(i)} &= \frac{\alpha_t^{g2,(i)}}{\alpha_t^{g1,(i)} + \alpha_t^{g2,(i)}}
 \end{aligned}$$

3. Merge

$$\begin{aligned}
 \hat{x}_{t|t}^{g,(i)} &= \bar{\alpha}_t^{g1,(i)} \hat{x}_{t|t}^{g1,(i)} + \bar{\alpha}_t^{g2,(i)} \hat{x}_{t|t}^{g2,(i)} \\
 P_{t|t}^{g,(i)} &= \bar{\alpha}_t^{g1,(i)} \left(P_{t|t}^{g1,(i)} + \left(\hat{x}_{t|t}^{g1,(i)} - \hat{x}_{t|t}^{g,(i)} \right)^2 \right) + \\
 &\quad \bar{\alpha}_t^{g2,(i)} \left(P_{t|t}^{g2,(i)} + \left(\hat{x}_{t|t}^{g2,(i)} - \hat{x}_{t|t}^{g,(i)} \right)^2 \right)
 \end{aligned}$$

4. Time update

$$\begin{aligned}
 \hat{x}_{t+1|t}^{g,(i)} &= \hat{x}_{t|t}^{g,(i)} \\
 P_{t+1|t}^{g,(i)} &= P_{t|t}^{g,(i)} + G_t^g Q_t^g G_t^{gT}
 \end{aligned}$$

Particle filter

1. Weights

$$w_t^{(i)} = \alpha_t^{g1,(i)} + \alpha_t^{g2,(i)}$$

$$\bar{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^N w_t^{(j)}}$$

2. Resampling

If $N_{\text{eff}} < N_{\text{th}}$, e.g., $\frac{1}{\sum_{i=1}^N \bar{w}_t^{(i)}} < N_{\text{th}}$ then resample

and set $\bar{w}_t^{(i)} = \frac{1}{N}$

3. Sample

For $i = 1, \dots, N$, sample $x_{t+1}^{p,(i)} \sim p(x_{t+1}^p | x_t^{p,(i)})$ where

$$p(x_{t+1}^p | x_t^{p,(i)}) =$$

$$\mathcal{N}\left(F_{p,t}^p x_t^{p,(i)} + F_{k,t}^p \hat{x}_{t|t-1}^{k,(i)}, Q^{add} + F_{k,t}^p P_{t|t-1}^k F_{k,t}^{p\ T}\right)$$

Kalman filter

1. Measurement update

$$\hat{x}_{t|t}^{k,(i)} = \hat{x}_{t|t-1}^{k,(i)} + P_{t|t-1}^k F_{k,t}^{p\ T} S_t^{k-1} \left(y_t^{k,(i)} - F_{k,t}^p \hat{x}_{t|t-1}^{k,(i)} \right)$$

$$P_{t|t}^k = P_{t|t-1}^k - P_{t|t-1}^k F_{k,t}^{p\ T} S_t^{k-1} F_{k,t}^p P_{t|t-1}^k$$

$$S_t^k = F_{k,t}^p P_{t|t-1}^k F_{k,t}^{p\ T} + Q^{add}$$

2. Time update

$$\hat{x}_{t+1|t}^{k,(i)} = F_{p,t}^k x_t^{p,(i)} + F_{k,t}^k \hat{x}_{t|t}^{k,(i)}$$

$$P_{t+1|t}^k = F_{k,t}^k P_{t|t}^k F_{k,t}^{k\ T} + G_t^k Q_t^k G_t^{k\ T}$$

Output

1. State estimation

$$\hat{x}_t^p = \sum_{i=1}^N \bar{w}_t^{(i)} x_t^{p,(i)}$$

$$\hat{x}_t^g = \sum_{i=1}^N \bar{w}_t^{(i)} \hat{x}_{t|t}^{g,(i)}$$

$$\hat{x}_t^k = \sum_{i=1}^N \bar{w}_t^{(i)} \hat{x}_{t|t}^{k,(i)}$$

2. Covariance estimation

$$P_t^p = \sum_{i=1}^N \bar{w}_t^{(i)} \left(x_t^{p,(i)} - \hat{x}_t^p \right) \left(x_t^{p,(i)} - \hat{x}_t^p \right)^T$$

$$P_t^g = \sum_{i=1}^N \bar{w}_t^{(i)} \left(P_{t|t}^{g,(i)} + \left(\hat{x}_{t|t}^{g,(i)} - \hat{x}_t^g \right)^2 \right)$$

$$P_t^k = P_{t|t}^k + \sum_{i=1}^N \bar{w}_t^{(i)} \left(\hat{x}_{t|t}^{k,(i)} - \hat{x}_t^k \right) \left(\hat{x}_{t|t}^{k,(i)} - \hat{x}_t^k \right)^T$$

Chapter 5

Simulations

In this chapter the results from the simulations will be presented. Two flights have been used to illustrate the strengths and the weaknesses of this algorithm. Two alternative methods to deal with the weaknesses will be discussed last in this chapter. Flight data recorded during real flights were used for the simulations. This means that the algorithm is tested on authentic data, and the results obtained are the same as they would have been for a real flight. Due to the stochastic nature of the algorithm, the simulations were performed several times with the same data. The result is a mean of these repeated simulations and can be used to find confidence intervals for the estimations.

This chapter will present the flights, discuss the design parameters, motivate the evaluation methods and finally show the results. First of the result sections is the algorithm as described in Chapter 4. Then there are two sections describing the result using two methods to deal with the weaknesses. Last is a part that compares the NINS algorithm with the one presented here.

First of all we will see what is really achieved by this algorithm. Figure 5.1 shows the position estimation error for the INS alone and for the sensor fusion described in Chapter 4. As can be seen the INS alone drifts constantly while the result obtained from the algorithm is stable.

5.1 Flights

To be able to evaluate the algorithm, the simulations were run on two different sets of data. One set is from a flight where the aircraft is close enough to the ground to get reliable data from the radar altimeter all the time. The terrain contains lots of information most of the time, but also have intervals of less information as it passes over sea. This flight should be an easy flight for the algorithm and will give some kind of

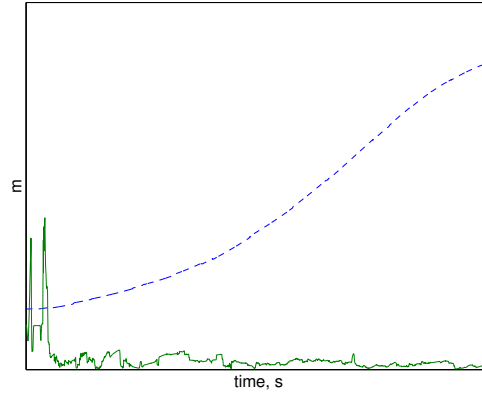


Figure 5.1: Plot of the position estimation error for one of the used flights. The INS alone (dashed) and the sensor fusion described in Chapter 4 (solid).

best performance measure of the algorithm. The flight path is shown in Figure 5.2.

The second flight has been chosen to test the algorithm when data is missing. Some periods of this flight the aircraft is at an altitude that makes the radar altimeter data unreliable, and the algorithm will therefore consider it as a lack of data. Some periods, the aircraft engages long turns, which also make the data unreliable. The test is to see if the algorithm can recover after the lack of data. The flight path is shown in Figure 5.3.

5.2 Parameters

There are lots of parameters that have to be chosen. The number of samples, initial values for the particle filter, process noise, added noise in horizontal position to compensate for the discretization in the particle filter and measurement noise are the design parameters that will be addressed in the next few sections. Qualitatively estimations and some trial-and-error have given the parameters.

5.2.1 Number of samples

The number of samples decides how fast the algorithm will converge and how well it will estimate the state, see Figure 5.4. By doubling the number of samples the time to convergence is typically halved and also the error is a lot smaller. This is true up to about 5000 samples for

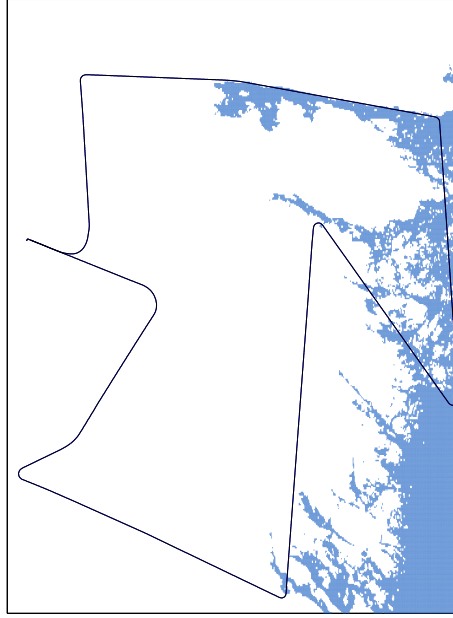


Figure 5.2: Plot of the path for the first flight. The dark regions on the right is the Baltic Sea to the east of Sweden. The track is clockwise and while passing the sea there is very little terrain information available for the algorithm.

this application, where it seems to level out. In the simulations made in this chapter the number of samples will be 5000.

5.2.2 Initial values

For x^g and x^k it is easy since they are estimated by Kalman filters, their initial values are set to zero. Initial covariance for x^g and x^k are taken from Nordlund [14] and are presented in Table 5.1. How about the initial values for the particle filter samples? The simplest way is to sample the initial samples uniformly in a rectangle. Since the initial position of the aircraft is approximately known before take off this rectangle does not have to be that large. The sides of the rectangle have been chosen to make the standard deviation equal to 1 km in both directions.



Figure 5.3: Plot of the path for the second flight. This flight is also clockwise. The dark regions of the map corresponds to the Baltic Sea and the lake Mälaren.

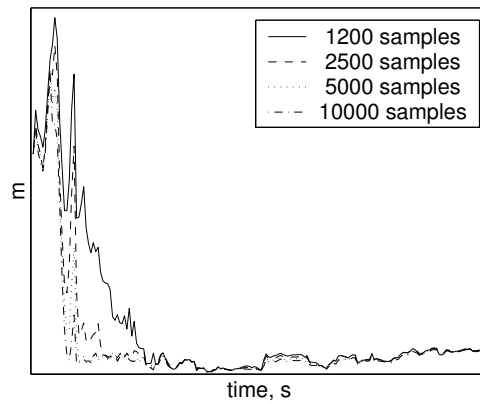


Figure 5.4: This plot shows the estimation error in the horizontal position as time evolves for different number of samples.

5.2.3 Process noise

Q^{add} controls the spread of the samples, and the amount of information in the height database in the area determines how well the algorithm concentrates the samples again. These two factors will together create a balance, and it is by Q^{add} that we control the size of the sample cloud for this balance. A smaller Q^{add} might cause the algorithm to lose the real state and drift blindly. A bigger Q^{add} will instead cause the algorithm to become less concentrated in the interesting region with less accuracy as a result. When missing data the cloud will spread too fast if Q^{add} is large. The estimated covariance of the particle filter, P_t^p , is used to calculate Q^{add} at each iteration

$$Q_t^{add} = c^{add} P_t^p, \quad (5.1)$$

where c^{add} is a constant that depends on the number of samples and is typically chosen to be 10^{-3} for 5000 samples.

The process noise for x^g and x^k are taken from Nordlund [14] and is set to

$$Q_t = \begin{bmatrix} Q_t^g & 0 \\ 0 & Q_t^k \end{bmatrix} = \text{diag} (0.2, 10^{-4}, 10^{-4}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6})^2. \quad (5.2)$$

In Nordlund [14] these values are used to simulate the process noise. Here real flight data is used but the values seem to be sufficient enough. They can be approximated by studying the \mathcal{O} -terms in the linearization in Chapter 2.

5.2.4 Measurement noise

As described in Section 4.3 the radar altimeter has a measurement noise that is a Gaussian mixture. The radar altimeter has one lobe directed straight down from the belly of the aircraft. Any echo is recognized as the ground. Sometimes the beam is reflected off the tree tops, and the radar altimeter will give a false measurement. This can be modelled by two Gaussians according to Figure 5.5, and an expression for the noise may be written as

$$p_{e_t^{rhm}} = \pi_1 \mathcal{N}(m_1, \sigma_1) + \pi_2 \mathcal{N}(m_2, \sigma_2) \quad (5.3)$$

where π_1 and π_2 are the probabilities of reading an echo from the ground or from the tree tops respectively. The easiest way to deal with this problem, is to estimate this noise as a singular Gaussian and use a Kalman filter. Fortunately there is the GPB filter, described in Section 2.2, that can use the model with two Gaussians. The performance

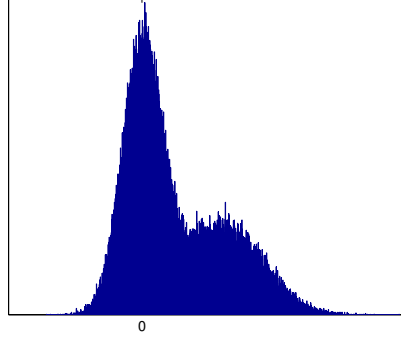


Figure 5.5: Typical histogram of the error in the data from the radar altimeter. The first bump corresponds to readings of the ground and the other to readings from the tree tops.

difference is large enough to motivate the slightly more complicated GPB filter. See Nordlund [13] for a comparison of performance.

When the aircraft is turning, it might lean to much for the radar altimeter to be directed downwards. The radar lobe is quite wide but if either the roll or the pitch is too large the measurements will be unreliable. Also when the aircraft is at high altitude, the measurement might be unreliable. This may be caused by an echo from the last pulse sent that is first registered as an echo for the next pulse.

In the measurement equation there is also the height database. This is a commercial database that has a height for positions in a grid where the points are separated by 50 m. In between the grid points the height is obtained by bilinear interpolation. This means that there is error in the measurement equation due to the interpolation and the accuracy of the database entries. In this chapter this has been modelled as an additional covariance in the measurement noise, i.e., the measurement noise can be written,

$$p_{e_t} = \pi_1 \mathcal{N}(m_1, \sigma_1 + R) + \pi_2 \mathcal{N}(m_2, \sigma_2 + R) \quad (5.4)$$

where R is the added covariance. In this chapter the added covariance is constant $R = 10^2$ in the first result section. In the last sections in this chapter there are discussions and results of tests with variable covariance.

5.2.5 Parameter summary

Here is a summary of the parameters used and their values.

Table 5.1: Parameters used in the algorithm

N	5000
$p(x_0^p)$	$\mathcal{U}\left(-\frac{1000\sqrt{3}}{r_0}, \frac{1000\sqrt{3}}{r_0}\right) \cdot \mathcal{U}\left(-\frac{1000\sqrt{3}}{r_0 \cos L_0}, \frac{1000\sqrt{3}}{r_0 \cos L_0}\right)$
$p(x_0^g, x_0^k)$	$\mathcal{N}(0, P_0)$
$P_0 = \text{diag}$	$(50, 1, 1, \frac{0.05\pi}{180}, \frac{0.05\pi}{180}, \frac{0.1\pi}{180}, 5 \cdot 10^{-3}, 5 \cdot 10^{-3})^2$
$p(u_t)$	$\mathcal{N}(0, Q_t)$
$Q_t = \text{diag}$	$(0.2, 10^{-4}, 10^{-4}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6})^2$
R	10^2 in the first result section
N_{th}	$\frac{2N}{3}$

5.3 Evaluation methods

To get a statistically valid result, the tests were run 100 times each. From these runs the root mean square error, $RMSE_t$, and the mean of the covariance, P_t^{mean} , were calculated according to

$$RMSE_t = \left(\frac{1}{M} \sum_{i=1}^M \left\| \hat{x}_t^{(i)} - x_t^{true} \right\|_2^2 \right)^{\frac{1}{2}} \quad (5.5a)$$

$$P_t^{mean} = \frac{1}{M} \sum_{i=1}^M P_t^{(i)}, \quad (5.5b)$$

where M is the number of runs. The true states at time t are obtained from the differential GPS (DGPS). This system uses data from the GPS satellites along with a correction signal from receivers with known positions. The problem is that we do not have the true values for the attitude and the accelerometer biases. The behavior of P_t^{mean} is very important, since it indicates how confident the algorithm is. The square root of P_t^{mean} is called the standard deviation. Mathematical statistics states that the $RMSE_t$ should be less than the standard deviation in 67% of the time, in a Gaussian case. If the model is correct the $RMSE_t$ and the standard deviation should follow that result approximately in these simulations, but as we will see, the $RMSE$ is often a lot better than that. This is a result of the added process noise. Eventhough the standard deviation of the algorithm increases with the added process noise, the $RMSE_t$ is smaller. Simulation runs, where the added noise is smaller, shows that the model is in fact fairly correct.

In a real application the standard deviation is important as an indicator, since there is no available correct value. A statistical upper limit of the real estimation error can be calculated from P_t^{mean} , by knowing the characteristics of the standard deviation.

5.4 Constant map covariance

In this section we will see the result from the test when the added covariance due to the inaccuracy in the map is constant.

5.4.1 Results

First we will study the estimation error in the horizontal position. A plot of the $RMSE_t$ and the square root of P_t^{mean} for the position can be found in Figure 5.6. It is easily seen that the algorithm is not getting enough information during the time when it passes the sea. The $RMSE_t$ during this period is not growing in the same rate as P_t^{mean} , but there were some simulation runs where the two follow each other more closely than in this plot. Overall the algorithm performs well and notice that it seems to know how accurate it is.

In the beginning of the simulation the algorithm is trusting the measurements a little bit too much. The result of that is the quite violent leaps in the $RMSE_t$. One method to deal with just that is described in Section 5.5.

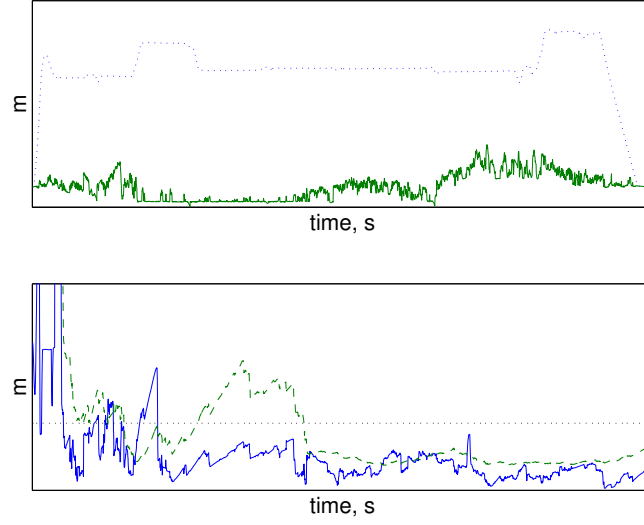


Figure 5.6: The top plot is the altitude profile of the aircraft (dotted) and of the ground elevation (solid). The bottom plot is the position estimation error $RMSE_t$ (solid), $\sqrt{P_t^{mean}}$ (dashed) and a reference level (dotted). Constant covariance for the map inaccuracy is used and the data is from the first flight.

Simulations with data from the second flight shows properties of the algorithm when there is a lack of data. See Figure 5.7. When the aircraft is at high altitude there are no reliable measurements available. We can see that the algorithm drifts when the altitude is high. When the data is back again it takes a little while for the algorithm to gather the samples again. As long as it has got some time it will converge to an estimation error that is good enough. In the end of the flight however the lack of data is due to the sharp turns. It seems there is not enough time for the algorithm to converge in between the turns, and in the end of the flight it has lost the position and is drifting.

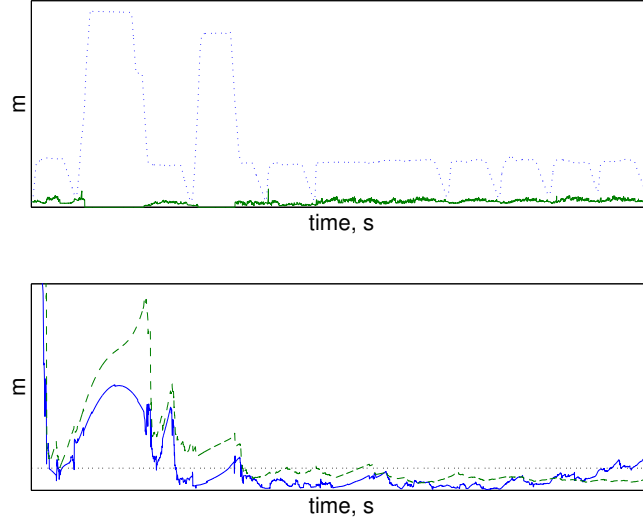


Figure 5.7: The top plot is the altitude profile of the aircraft (dotted) and of the ground elevation (solid). The bottom plot is the position estimation error $RMSE_t$ (solid), $\sqrt{P_t^{mean}}$ (dashed) and a reference level (dotted). Constant covariance for the map inaccuracy is used and the simulation data is from the second flight.

Unfortunately the accuracy of the DGPS is not very good in the altitude channel. In Figure 5.8 we can see the results of the estimation in the altitude channel. The covariance is stable at a low level while the estimation error is not as good. Considering the good results in horizontal position, we should be careful with the conclusions. In this case we will have to trust the algorithm and its P_t^{mean} .

The standard deviation, $\sqrt{P_t^{mean}}$, converge fast and is stable for the entire flight, while the mean of the $RMSE_t$ is considerably larger. Assuming the model of the radar altimeter is correct, we can estimate the error in the DGPS. The altitude estimation error, when compensating for the bias in the DGPS, is shown in Figure 5.9. This plot might be more correct, but that conclusion can not be proved. It should therefore only be considered as an illustrative plot.

The estimation of the velocity is very important, since it controls the drifting of the algorithm during the lack of data. In Figure 5.10 we see that the $RMSE_t$ and the P_t^{mean} are both stable. The peaks in the plot corresponds to the fact that it takes a while for the DGPS to realize that the direction of the velocity has changed. DGPS differentiates the position data to obtain the velocity.

In the plots we can see that the algorithm converge quite quickly for

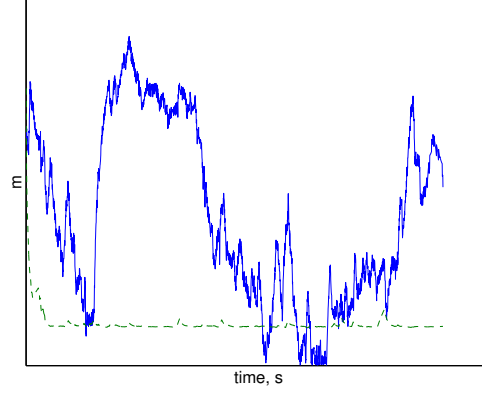


Figure 5.8: The estimation error $RMSE_t$ (solid) and the square root of P_t^{mean} (dashed) of the altitude error. Constant covariance for the map inaccuracy is used and the simulation data is from the first flight.

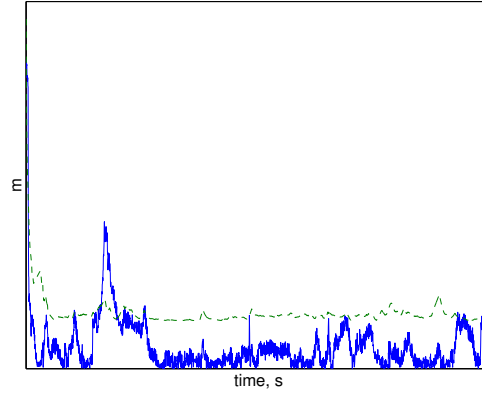


Figure 5.9: The estimation error $RMSE_t$ (solid), compensated for the DPGS error, and the square root of P_t^{mean} (dashed) of the altitude error. Constant covariance for the map inaccuracy is used and the simulation data is from the first flight.

the horizontal position and the altitude. This may be enough, as long as the algorithm gets new observations. When the aircraft is landing there is often less information or none at all the last minute, due to the terrain or that the aircraft is too close to the ground to get reliable measurements. Therefore it is important that even the velocity has

converged. It is first when the position and altitude has converged, that the velocity estimation can really begin.

Results for the position, altitude and velocity are enough to illuminate the overall result of the algorithm why results for attitude, heading and accelerometer biases will not be further commented.

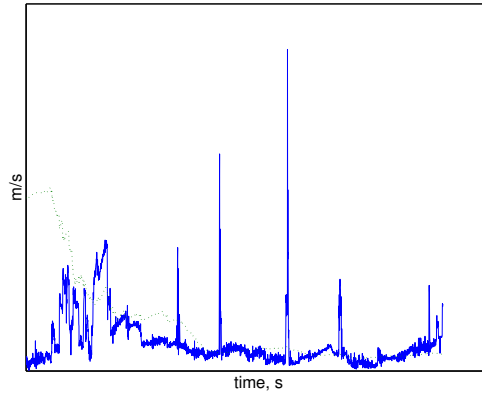


Figure 5.10: The estimation error $RMSE_t$ (solid) and the square root of P_t^{mean} (dashed) of the velocity error. Constant covariance for the map inaccuracy is used and the simulation data is from the first flight.

5.5 Variable map covariance

Consider for example the case when flying over sea. In this case the bilinear interpolation is quite accurate. Consider instead passing a very rough terrain where the interpolation introduce significant error. It seems that adding a constant covariance for the map inaccuracy is not the optimal way. In this section we will introduce variable covariance for the inaccuracy of the map. This is done by placing a grid of $500\text{m} \times 500\text{m}$ around the estimated position and calculating the covariance of the database heights at the grid points. This is then added to the fixed covariance R . In the tests $R = 8^2$.

5.5.1 Results

The main difference from the case where the covariance was constant, is the smoother beginning and better accuracy. This can be seen in Figure 5.11. The algorithm now tries to suppress the impact of measurements, where the terrain is very rough. Before, the impact on the

weight vector from such measurements were significant enough to make the algorithm discard good samples close to the real state. During the interval where there is most differences from the result with constant map covariance, the mean of the $RMSE_t$ is reduced by almost 40%. Throughout the simulation there are fewer and smaller leaps when using variable map covariance.

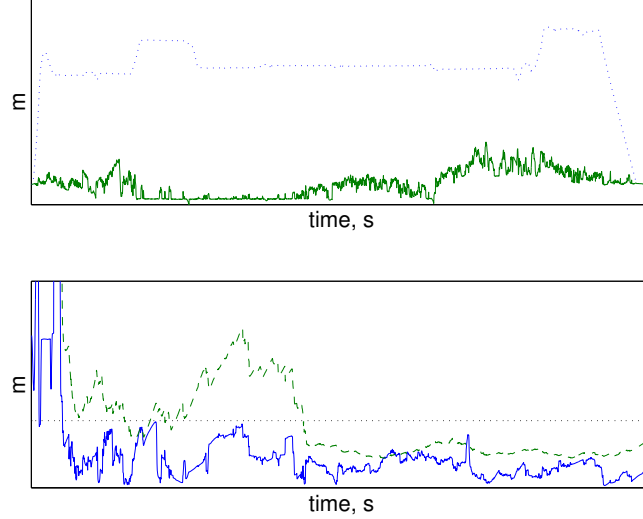


Figure 5.11: The top plot is the altitude profile of the aircraft (dotted) and of the ground elevation (solid). The bottom plot is the position estimation error $RMSE_t$ (solid), $\sqrt{P_t^{mean}}$ (dashed) and a reference level (dotted). Variable covariance for the map inaccuracy is used and the data is from the first flight.

5.6 Variable added process noise

The main reason for adding process noise into the particle filter, is to get the samples to move within the sample cloud. Otherwise, if the terrain is very rough some samples may be close to the real states but still have bad agreement with the measurements. The undesired result of this added noise, is that the cloud tends to grow more than necessary. Most of the time the algorithm can concentrate the samples by resampling, but for that the samples need to have differences in the weights. If the terrain has little information, i.e., the weight vector is nearly constant, the algorithm can not concentrate the sample cloud. Instead the cloud grows uncontrolled.

In the preceeding sections the added process noise have been constant. Here, the idea is to keep the estimated covariance constant by using variable added noise. This will make the algorithm use more added noise when the algorithm can concentrate the samples effectively and less when there is little information. It is not necessary to move the samples if the contribution to the weight vector is the same. By using an upper limit the algorithm is allowed to concentrate the sample cloud when there is lots of information. Before we get to the details we need a way to determine if the estimated covariance will increase or decrease with the information at hand.

5.6.1 Cramer-Rao bounds

For every iteration there is, in this case, a lower bound for the covariance according to

$$(P_t^{CR})^2 = Q_t^{add} Z_t^{-1} R = c_t^{add} P_t^p Z_t^{-1} R, \quad (5.6)$$

where P_t^{CR} is the Cramer-Rao lower bound for the covariance. Z_t is a measurement of the amount of information at the current position. This is calculated as a 2 by 2 matrix of differentials of the height database in north and east directions

$$Z_t = E \left[\left(\begin{array}{c} \frac{\partial h(x)}{\partial x_n} \\ \frac{\partial h(x)}{\partial x_e} \end{array} \right) \left(\begin{array}{cc} \frac{\partial h(x)}{\partial x_n} & \frac{\partial h(x)}{\partial x_e} \end{array} \right)^T \right] \bigg|_{x=x_t^p}. \quad (5.7)$$

Z_t can be calculated as the mean of the differentials at each sample. In this thesis the same grid as was used for the variance of the height database, is used to calculate Z_t .

The added noise is calculated so that $P_t^{CR} < P_t^p$. Of course this cannot always be done, since the real process noise is still there when the added noise approaches zero. To calculate the desired c_t^{add} we try to make $P_t^{CR} = P_t^p$. Rewrite Equation (5.6) as

$$(P_t^p)^2 = c_t^{add} P_t^p Z_t^{-1} R. \quad (5.8)$$

From this we can get a value for c_t^{add} to ensure that the estimate covariance is not increased more than necessary. The design parameter c^{add} is now used as an upper bound for c_t^{add} . When $c_t^{add} > c^{add}$, it means that there are information enough to decrease the estimate covariance, even though the maximum added noise is used.

5.6.2 Results

As could be seen in Section 5.4.1 the algorithm seems to lose the real state and starts drifting in the end of the second flight. This may be

prevented by using more added noise. During the periods when the algorithm lacks data, this would make the cloud grow too much. When using variable added noise, we can use a higher c^{add} . The result is shown in Figure 5.12. In the end the algorithm still keeps track of the position. During the flight the estimated covariance is bigger than it was before, due to that the added noise is now 2.5 times bigger. From the time where the algorithm lost track of the real position, the mean of the $RMSE_t$ is here reduced by more than 40%. Here the fact that the algorithm still keeps track of the real position and keeps the $RMSE_t$ as low as it does is important. Earlier the algorithm lost track and drifted, which means that the mean $RMSE_t$ would increase uncontrolled if the flight continued.

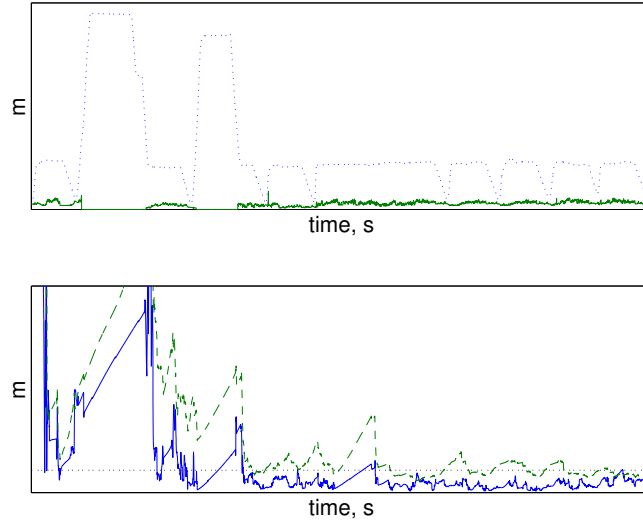


Figure 5.12: The top plot is the altitude profile of the aircraft (dotted) and of the ground elevation (solid). The bottom plot is the position estimation error $RMSE_t$ (solid), $\sqrt{P_t^{mean}}$ (dashed) and a reference level (dotted). Variable covariance for the map inaccuracy and variable added noise is used and the simulation data is from the second flight.

5.7 Comparison to existing methods

Today NINS is running with the INS, TAP and GPS. This means that it is not fair to make comparisons between NINS and the algorithm presented in this thesis. There have been test flights where only INS and RHM were

used and it is the second flight. Therefore there will be only one plot of the performance of these two. As can be seen in Figure 5.13 the two algorithms are quite similar in performance. Please keep in mind that the algorithm in this thesis has only been tried on real flight data during this master thesis project, and there are many adjustments that would surely make it better. Figure 5.14 is a plot of the altitude comparison and Figure 5.15 is a plot of the velocity comparison. As can be seen the two algorithms are very similar in performance. The altitude plot is again compensated for the error in DGPS and should only be seen as a illustrative plot.

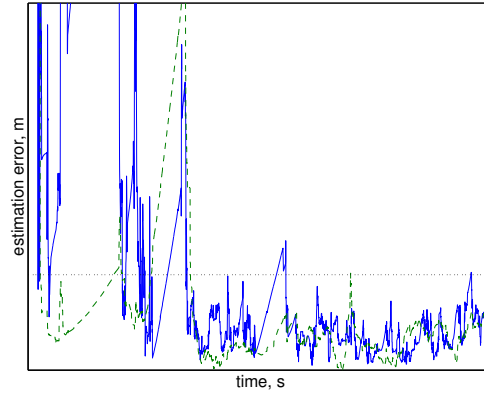


Figure 5.13: The estimation error of the algorithm presented in this thesis (solid), the estimation error of NINS, using only INS and RHM (dashed) and a reference level (dotted).

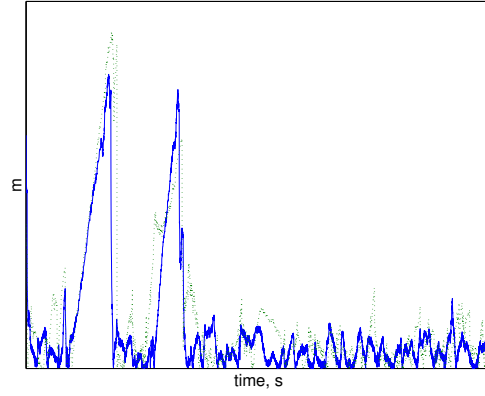


Figure 5.14: The altitude estimation error of the algorithm presented in this thesis (solid) and the estimation error of NINS, using only INS and RHM (dotted). With compensation for the error in DGPS altitude.

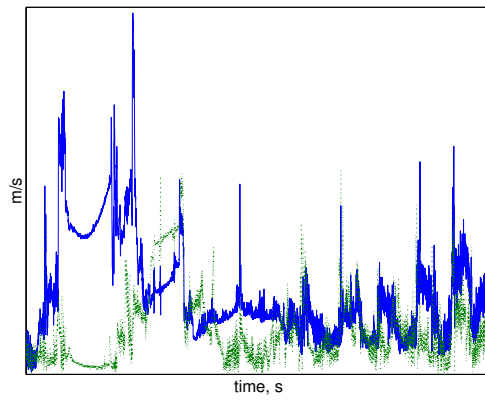


Figure 5.15: The velocity estimation error of the algorithm presented in this thesis (solid) and the estimation error of NINS, using only INS and RHM (dotted).

Chapter 6

Conclusions and future work

In this thesis a new sensor fusion algorithm have been tested using real flight data. The results in Chapter 5 indicate the potentials and the weaknesses. How these potentials can be used, and how to deal with the weaknesses are the topics of this last chapter.

6.1 Conclusion

All the plots in Chapter 5 give an indication of the potential of this algorithm. The comparison with the NINS algorithm showed very similar performance. Please remember that NINS uses some techniques to use more data, and that it has been developed for a few years. The algorithm presented in this thesis has only been tested on real flight data during this thesis.

This non-linear sensor fusion works very well as long as there is reliable data. It even meets the requirements for NINS. To make it work properly, it needs quite some time with lots of terrain information to converge in all states. For landing purposes, about half the time is in most cases enough.

For a real implementation of this algorithm, computation requirements are of great interest. The simulations done for this thesis has been running in real time using a SUN station 900 Mhz. About 90% of the time is used for height database queries. The work load on the system computer onboard an aircraft is therefore very hard to guess. It all depends on the time required to perform database queries.

6.2 Future work

There is still lots of things to do before this algorithm is ready for system implementation. First there should be some more studies on the robustness issue, and the work load on a system computer. In Chapter 5 there are two techniques used to deal with the weaknesses. These are only ideas and there is some theory to study behind them. Except the robustness issue, the handling of lack of data is probably the single most important issue for better performance.

References

- [1] C Andrieu, A Doucet and W J Fitzgerald. An Introduction to Monte Carlo Methods for Bayesian Data Analysis. Technical report, Department of Engineering, University of Cambridge, CB2 1PZ Cambridge, UK.
- [2] N Bergman. *Recursive Bayesian Estimation Navigation and Tracking Applications*. PhD thesis no. 579, Department of Electrical Engineering, Linköpings universitet, 1999.
- [3] H A P Blom, Y Bar-Shalom. The Interacting Multiple Model Algorithm for Systems with Markovian Switching Coefficients. *IEEE Transactions on Automatic Control*, AC-33(8):780–783, August 1988.
- [4] R Chen, J S Liu. Mixture Kalman filters. *J. R. Statist. Soc. B*, (62, Part 3):493–508, 2000.
- [5] A Doucet, N Freitas, K Murphy, S Russel. Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks. Technical report, Engineering Dept., Cambridge University, UK.
- [6] A Doucet, N J Gordon, V Krishnamurthy. Particle Filters for State Estimation of Jump Markov Linear Systems. Technical Report CUED/F-INFENG/TR 359, Department of Engineering, University of Cambridge, CB2 1PZ Cambridge, UK, 1999.
- [7] N J Gordon, D J Salmond, A F M Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *IEE Proceedings-F*, (2):107–113, April 1993.
- [8] F Gustafsson, L Ljung and M Millnert. *Signalbehandling*. Studentlitteratur, Lund, Sweden, 2nd edition, 2001. In swedish.
- [9] A H Jazwinski. *Stochastic processes and filtering theory*. Academic Press, 1970.

-
- [10] R Karlsson, F Gustafsson, T Karlsson. Particle Filtering and Cramer-Rao Lower Bound for Underwater Navigation. Technical Report LiTH-isy-R-2474, Department of Electrical Engineering, Linköpings universitet, SE-581 83 Linköping, Sweden, October 2002.
 - [11] National Imagery and Mapping Agency. World Geodetic System 1984. Technical Report TR 8350.2, Department of Defence, USA, January 2000. 3rd edition.
 - [12] P-J Nordlund. Recursive state estimation of nonlinear systems with applications to integrated navigation. Technical Report LiTH-isy-R-2321, Department of Electrical Engineering, Linköpings universitet, SE-583 81 Linköping, Sweden, November 2000.
 - [13] P-J Nordlund. Recursive estimation of three-dimensional aircraft position using terrain-aided positioning. Technical Report LiTH-isy-R-2403, Department of Electrical Engineering, Linköpings universitet, SE-581 83 Linköping, Sweden, November 2001.
 - [14] P-J Nordlund. *Sequential Monte Carlo Filters and Integrated Navigation*. Lic thesis 945, liu-tek-lic-2002:18, Department of Electrical Engineering, Linköpings universitet, SE-581 83 Linköping, Sweden, 2002.
 - [15] W J Rugh. *Linear System Theory*. Prentice Hall, Upper Saddle River, N.J. USA, 2nd edition, 1996.

Notation

Model variables and functions

e_t	Measurement noise in the state space equations
f_t	Non-linear function, state transition
F_t	Linear function, state transition matrix
G_t	Linear function, process noise propagation matrix
$h(\cdot)$	Ground elevation database
h_t	Non-linear function, state to measurement function
H_t	Linear function, state to measurement matrix
N	Number of samples in the particle filter
N_{eff}	Effective sample size, $N_{\text{eff}} < N$
N_{th}	Threshold for N_{eff} for resampling
Q_t	Process noise covariance matrix
P_t	State covariance matrix
$P_{t t}, P_{t-1 t}$	State covariance matrices from the Kalman filter
$\hat{P}(X_t Y_t)$	Discrete estimation of $p(X_t Y_t)$
R_t	Measurement noise covariance matrix
T_s	Sampling period
u_t	Process noise in the state space equations
w_t, \bar{w}_t	Particle filter weights
x_t	State at time t , discrete
x_t^{INS}	State vector from the INS
\tilde{x}_t	State error vector
X_t	Stacked state vector, $X_t = \{x_0, \dots, x_t\}$
$\hat{x}_{t t}, \hat{x}_{t-1 t}$	State estimation from a Kalman filter
y_t	Measurement at time t
Y_t	Stacked measurement vector, $Y_t = \{y_0, \dots, y_t\}$
$\alpha_t, \bar{\alpha}_t$	GPB filter weights
λ_t	GPB mode at time t
Λ_t	Stacked mode vector, $\Lambda_t = \{\lambda_0, \dots, \lambda_t\}$
Λ_{t-L+1}^t	Stacked mode vector, $\Lambda_{t-L+1}^t = \{\lambda_{t-L+1}, \dots, \lambda_t\}$
π_{kl}	Mode transition probability, $\pi_{kl} = p(\lambda_t = k \lambda_{t-1} = l)$

Navigation variables

a^b	Specific acceleration in body frame
b^a	Accelerometer bias vector
b_x^a, b_y^a	Components of accelerometer bias in body frame
C_b^n	Conversion matrix, body frame to navigation frame
f^b	Specific acceleration including gravity
h	Altitude
L	Latitude
l	Longitude
v^n	Velocity in navigation frame
v_n, v_e, v_d	Components of velocity in navigation frame
ϕ	Roll
Γ^n, γ^n	Small angle transformation matrix and vector
$\gamma_n, \gamma_e, \gamma_d$	Components of the small angle transformation vector
θ	Pitch
Ω, ω	Angular velocity in skew-symmetric form and vector form
ψ	Heading

Constants

e	Earth's ellipticity, WGS84
g_0	Gravity constant, WGS84
J	Empirical constant, WGS84
r_0	Semimajor axis, mean equator radius, WGS84
ϵ^2	Earth's eccentricity, WGS84
ω_{ie}	Earth's angular velocity, WGS84

Abbreviations

DGPS	Differential GPS
GPS	Global Positioning System
INS	Inertial Navigation System
NINS	New Integrated Navigation System
RHM	Radar altimeter
TAP	Terrain Aided Positioning
P^{mean}	Mean of the estimated covariance
$RMSE$	Root mean square error

Appendix A

Continuous state space model

In Section 3.4.6 there is an expression of the continuous state space model. Here are the complete matrices $A(t)$ and $B(t)$.

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \quad (\text{A.1})$$

$$A(t) = \begin{bmatrix} 0 & 0 & 0 & \frac{1}{r_0} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{v_e \sin L}{r_0 \cos^2 L} & 0 & 0 & 0 & \frac{1}{r_0 \cos L} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_L^{v_n} & 0 & 0 & \frac{v_d}{r_0} & a_{v_e}^{v_n} & 0 & f_z & -f_e & C_{b,11}^n & C_{b,12}^n \\ a_L^{v_e} & 0 & 0 & a_{v_n}^{v_e} & a_{v_e}^{v_e} & -f_z & 0 & f_n & C_{b,21}^n & C_{b,22}^n \\ \omega_{ie} \sin L & 0 & 0 & 0 & -\frac{1}{r_0} & 0 & \omega_{in,d}^n & -\omega_{in,e}^n & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{r_0} & 0 & -\omega_{in,d}^n & 0 & \omega_{in,n}^n & 0 & 0 \\ a_L^{\gamma_d} & 0 & 0 & 0 & \frac{\tan L}{r_0} & \omega_{in,e}^n & -\omega_{in,n}^n & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (\text{A.2a})$$

$$a_L^{v_n} = -\frac{v_e^2}{r_0 \cos^2 L} - 2\omega_{ie} v_e \cos L \quad (\text{A.2b})$$

$$a_L^{v_e} = \frac{v_n v_e}{r_0 \cos^2 L} + 2\omega_{ie} v_n \cos L - 2\omega_{ie} v_d \sin L \quad (\text{A.2d})$$

$$a_{v_e}^{v_e} = \frac{v_n \tan L}{r_0} + \frac{v_d}{r_0} \quad (\text{A.2f})$$

[illegible]

Copyright

Svenska

Detta dokument hålls tillgängligt på Internet - eller dess framtida ersättare - under en längre tid från publiceringsdatum under förutsättning att inga extra-ordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart. För ytterligare information om *Linköping University Electronic Press* se förlagets hemsida <http://www.ep.liu.se/>

English

The publishers will keep this document online on the Internet - or its possible replacement - for a considerable time from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for your own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement. For additional information about the *Linköping University Electronic Press* and its procedures for publication and for assurance of document integrity, please refer to its WWW home page: <http://www.ep.liu.se/>

© Petter Frykman
Linköping, 28th April 2003

