Marginalized Particle Filter for Aircraft Navigation in 3-D
Marginalized Particle Filter for Aircraft Navigation in 3-D

Examensarbete utfört i Reglerteknik vid Tekniska högskolan i Linköping
av

Tomas Hektor

LITH-ISY-EX--07/4024--SE

Handledare: Per-Johan Nordlund
isy, Linköpings universitet
David Törnqvist
isy, Linköpings universitet
Henrik Karlsson
Saab AB

Examinator: Fredrik Gustafsson
isy, Linköpings universitet

Linköping, 24 October, 2007
Marginalized Particle Filter for Aircraft Navigation in 3-D

In this thesis Sequential Monte Carlo filters, or particle filters, are applied to aircraft navigation. This report consists of two parts. The first part is an illustration of the theory behind this thesis project. The second and most important part evaluates the algorithm by using real flight data.

Navigation is about determining one’s own position, orientation and velocity. The sensor fusion studied combines data from an inertial navigation system (INS) with measurements of the ground elevation below in order to form a terrain aided positioning system (TAP). The ground elevation measurements are compared with a height database. The height database is highly non-linear, which is why a marginalized particle filter (MPF) is used for the sensor fusion.

Tests have shown that the MPF delivers a stable and good estimate of the position, as long as it receives good data. A comparison with Saab’s NINS algorithm showed that the two algorithms perform quite similar, although NINS performs better when data is lacking.

Keywords: Inertial navigation, integrated navigation, marginalized particle filter, Rao-Blackwellization
Abstract

In this thesis Sequential Monte Carlo filters, or particle filters, applied to aircraft navigation is considered. This report consists of two parts. The first part is an illustration of the theory behind this thesis project. The second and most important part evaluates the algorithm by using real flight data.

Navigation is about determining one’s own position, orientation and velocity. The sensor fusion studied combines data from an inertial navigation system (INS) with measurements of the ground elevation below in order to form a terrain aided positioning system (TAP). The ground elevation measurements are compared with a height database. The height database is highly non-linear, which is why a marginalized particle filter (MPF) is used for the sensor fusion.

Tests have shown that the MPF delivers a stable and good estimate of the position, as long as it receives good data. A comparison with Saab’s NINS algorithm showed that the two algorithms perform quite similar, although NINS performs better when data is lacking.
Acknowledgments

First of all I would like to thank my supervisors, Per-Johan Nordlund and David Törnqvist at Linköpings Universitet who have helped me understand the theory, and Henrik Karlsson at Saab AB who has helped in many situations. I also want to thank my examiner professor Fredrik Gustafsson at Linköpings Universitet who has given helpful comments during the writing process.

Thanks to the people at the department of Primary Data and Navigation at Saab AB. You have been helpful, and you made me realize the importance of coffee breaks. Thanks also to my opponent Björn Benderius for highlighting areas that were not obvious to everyone.

Last but not least, thanks to my wife Emma for proofreading this thesis and motivating me throughout the whole project. I love you.
Contents

1 Introduction ............................................ 1
  1.1 Background ........................................ 1
  1.2 Specifications ..................................... 2
  1.3 Thesis Outline ..................................... 3

2 Integrated Aircraft Navigation ......................... 5
  2.1 Navigation Equations ............................... 5
  2.2 INS Error Dynamics ................................ 9
    2.2.1 Position Error Dynamics ..................... 9
    2.2.2 Velocity And Attitude Error Dynamics ........ 10
    2.2.3 Accelerometer Error Dynamics ............... 12
  2.3 Measurement Model ................................ 13
    2.3.1 Radar Error .................................. 13
    2.3.2 Map Uncertainty ............................. 14
  2.4 Summary Of Chapter 2 ............................. 14

3 State Estimation Background Theory .................. 15
  3.1 Kalman Filter ...................................... 15
  3.2 Multiple Models ................................... 16
  3.3 Recursive Bayesian Estimation .................... 18
  3.4 Particle Filter .................................... 19
    3.4.1 Importance Sampling .......................... 19
    3.4.2 Resampling ................................... 20
  3.5 Marginalized Particle Filter ....................... 22
  3.6 Extending The Marginalized Particle Filter ....... 24
  3.7 Cramér-Rao Lower Bound ........................... 27
  3.8 Summary Of Chapter 3 ............................. 28

4 Performance Evaluation On Test Flights ............... 29
  4.1 Filter Design ...................................... 29
    4.1.1 Number Of Particles ........................... 29
    4.1.2 Process Noise ................................ 29
    4.1.3 Measurement Noise ............................ 30
    4.1.4 Map Uncertainty ............................... 31
    4.1.5 Initial Values ................................ 32
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.6</td>
<td>Evaluation Methods</td>
<td>32</td>
</tr>
<tr>
<td>4.2</td>
<td>Flight Trajectories</td>
<td>33</td>
</tr>
<tr>
<td>4.3</td>
<td>Results</td>
<td>34</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Ordinary MPF - Extended MPF</td>
<td>34</td>
</tr>
<tr>
<td>4.3.2</td>
<td>MPF</td>
<td>36</td>
</tr>
<tr>
<td>4.3.3</td>
<td>NINS</td>
<td>37</td>
</tr>
<tr>
<td>4.4</td>
<td>Summary Of Chapter 4</td>
<td>38</td>
</tr>
<tr>
<td>5</td>
<td>Conclusions And Future Work</td>
<td>41</td>
</tr>
<tr>
<td>5.1</td>
<td>Conclusions</td>
<td>41</td>
</tr>
<tr>
<td>5.2</td>
<td>Future Work</td>
<td>41</td>
</tr>
</tbody>
</table>

**Bibliography** 43

**A Continuous State Space Model** 45
Notation

Abbreviations

CRLB Cramér-Rao Lower Bound
GPB Generalized Pseudo-Bayesian
GPS Global Positioning System
KF Kalman Filter
MPF Marginalized Particle Filter
NINS New Integrated Navigation System
PF Particle Filter
SITAN Sandia Inertial Terrain Aided Navigation
TERCOM Terrain contour matching
TERPROM Terrain profile matching

Symbols

\( e_t \) Measurement noise
\( f \) State equation transition mapping (discrete time)
\( F \) Linearized state update matrix
\( N \) Number of particles
\( N_{\text{eff}} \) Effective number of particles
\( N_{\text{th}} \) Threshold for effective number of particles
\( \mathcal{N}(\mu, P) \) Normal distribution with mean value \( \mu \) and covariance \( P \)
\( P_t \) Covariance matrix
\( p(\cdot) \) Probability density function
\( p(x_t|Y_t) \) Posterior density
\( p_e(\cdot) \) Measurement noise probability density
\( q(\cdot) \) Importance density
\( \sigma \) Standard deviation
\( T \) Sample period
\( u^{(i)} \) Weight for particle \( i \)
\( x_t \) State vector at time \( t \)
\( X_t \) Set of state vectors, \( X_t = \{x_0, x_1, \ldots, x_t\} \)
\( \hat{x}_{t|t} \) Estimate at time \( t \)
\( \hat{x}_{t+1|t} \) One step ahead prediction
\( x^{(i)}_t \) Sample of state vector
\( y_t \) Measurement at time \( t \)
\( Y_t \) Set of ordered measurements, \( Y_t = \{x_1, x_2, \ldots, x_t\} \)
Chapter 1

Introduction

1.1 Background

In the early days of aviation, pilots used a kind of terrain aided positioning (TAP) to navigate. By looking at the surroundings below them and comparing objects seen with a map, the pilots could determine their own position. This method of navigation did not always work. If the aircraft was at high altitude on a cloudy day the pilot could not see the ground at all – making navigation impossible.

In order to achieve more accurate and reliable navigation, different technical solutions have been developed. Aircraft of today almost always use an inertial navigation system (INS). The sensors in the INS measure the accelerations and angular rates in every direction, and with Sir Isaac Newton’s laws position and velocity can be computed. By integrating acceleration once we obtain linear and angular velocity, and by integrating twice we obtain position and orientation. Hence, if the initial position, velocity and orientation are known, the current state can be computed from the previous one. However, errors in the measurements propagate through the computations and are accumulated over time, and the INS errors need to be compensated for. An easy method to do this is using some kind of radio navigation, like the global positioning system (GPS). This implies dependency on data from outside the aircraft. Data that either could be jammed in warfare or be missing when reception is bad.

Military aircraft applications strive toward independence of information from outside the aircraft. This is why TAP still is an interesting alternative – the ground is very hard to "jam". However, rather than having a visual look at the ground TAP measures the terrain elevation profile below the aircraft and compares this profile with a height database. With a radar altimeter the ground clearance can be measured, and if the height above mean sea level is known, the terrain height can easily be estimated. This is illustrated in Figure 1.1.

TAP has been a field of research for many years. The most frequently referred algorithms are TERCOM (TERrain COntour Matching) and SITAN (Sandia Inertial Terrain Aided Navigation). Both algorithms originate from the 1970’s. TERCOM is batch oriented and correlates the terrain elevation profile with the database
periodically. SITAN is a sequential style system which uses a Kalman filter to update the aircraft’s state with every independent radar altimeter measurement. The most widely used algorithm for TAP in aircraft today is BAE’s TERPROM (TERrain PROfile Matching) system. According to TERPROM’s homepage the system is today used in the following aircraft: A-10, C-130, C-17, Eurofighter Typhoon, F-16, Harrier, Jaguar, Mirage 2000 and Tornado [15]. TERPROM uses a combination of TERCOM and SITAN. In Campbell [3] a brief survey on existing TAP systems is given.

1.2 Specifications

Saab AB has a system called New Integrated Navigation System (NINS) running today. This system, which have been under development since the late 70’s, uses INS and TAP with a Kalman filter to carry out the sensor fusion [13]. In order to obtain a good estimate of the aircraft’s position, the error from TAP needs to be modelled with good precision. The Kalman filter used in the sensor fusion requires linearized data from TAP. The error model from TAP, however, is non-linear and this introduces linearization errors.

More efficient computers have made Monte Carlo filtering techniques, such as the particle filter, tractable. Particle filters do not need a linearized problem and therefore performance should improve. This thesis will evaluate a sensor fusion carried out by an extended marginalized particle filter [12] with real flight data and compare it with the performance of NINS. In addition a comparison with an ordinary marginalized particle filter will be performed with respect to computational costs and performance.
1.3 Thesis Outline

This thesis consists of five chapters.

- Chapter 2 presents the equations for aircraft navigation and the state space model that will be used in this thesis.
- Chapter 3 covers briefly the theory behind the different parts of the sensor fusion.
- Chapter 4 presents a performance evaluation of the sensor fusion.
- Chapter 5 discusses some conclusions and make suggestions for future work.
Chapter 2

Integrated Aircraft Navigation

The purpose of this chapter is to present a state space model on the form

\[ \dot{x}_t = A(t)x(t) + B(t)u(t) \]  \hspace{1cm} (2.1)

for the inertial navigation system and a measurement model on the form

\[ y_t = h(x_t) + e_t \]  \hspace{1cm} (2.2)

for the radar altimeter. Chapter 3 then presents some methods to estimate the state in (2.1), of which the marginalized particle filter (MPF) is a promising approach. Chapter 4 then applies the MPF to the state space model presented in this chapter.

2.1 Navigation Equations

Inertial navigation systems (INS) are based upon the laws of classical mechanics formulated by Sir Isaac Newton. The INS measures linear and angular acceleration. If the initial position and velocity is known, acceleration can be integrated once to obtain velocity and twice to obtain position. In the same way the angular accelerations and an initial orientation gives the current orientation of the aircraft.

The orientation also provides the possibility to transform velocity and position into any appropriate coordinate system, relating the aircraft to earth.

The earth can be approximated with an ellipsoid which is flattened at the poles. The ellipsoid is characterized by its ellipticity, \( e \), and its semi-major axis, the mean radius of the earth at the equator, \( r_0 \). The most common reference ellipsoid today is the World Geodetic System 1984 (WGS84), and this is the ellipsoid which will be used throughout this thesis. The numerical values of this ellipsoid can be found in Table 2.1. The numerical values together with more information on WGS84 can also be found in [9].
### Table 2.1. Parameters for WGS84

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Notation</th>
<th>Numerical value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semi-major axis</td>
<td>( r_0 )</td>
<td>( 6.378137 \cdot 10^6 ) m</td>
</tr>
<tr>
<td>Earth’s ellipticity</td>
<td>( e )</td>
<td>( 1/298.2572 )</td>
</tr>
<tr>
<td>Angular velocity of the earth</td>
<td>( \omega_{ie} )</td>
<td>( 7.292115 \cdot 10^{-5} ) rad/sec</td>
</tr>
<tr>
<td>Gravity at equator</td>
<td>( g_0 )</td>
<td>( 9.780325 ) m/s(^2)</td>
</tr>
<tr>
<td>Gravity formula constant</td>
<td>( k_1 )</td>
<td>( 0.001931853 )</td>
</tr>
<tr>
<td>Gravity formula constant</td>
<td>( k_2 )</td>
<td>( 0.006802598 )</td>
</tr>
</tbody>
</table>

In order to characterize the INS mathematically we need a number of coordinate frames given by:

- **i** Inertial frame, fixed in the inertial space. For navigation periods shorter than days this frame can be approximated with an earth-centered, non-rotating frame.

- **e** Earth-centered frame, fixed to the earth, i.e. rotates with the earth.

- **n** Navigation frame, with its center attached to the aircraft. The \( x \), \( y \) and \( z \)-axis are aligned with north, east and the ellipsoid normal respectively. The velocity e.g. is denoted by \( v^n = [v_n, v_e, v_d]^T \).

- **b** Body frame, attached to the aircraft, i.e. translating and rotating with the aircraft. The \( x \), \( y \) and \( z \)-axis points through the nose, right wing and belly respectively. The velocity e.g. is denoted by \( v^b = [v_x, v_y, v_z]^T \).

The navigation equations below are merely presented in this thesis. More details can be found in e.g. [9] or [11].

Horizontal position is often given as two angles, latitude and longitude. Latitude is the angle between the equatorial plane and the normal to the ellipsoid. Longitude is the angle between the same normal and the plane which intersects the Greenwich meridian. Altitude is the height above the ellipsoid. Latitude, longitude and altitude will be denoted by \( L, l \) and \( h \) respectively, which are illustrated in figure 2.1. Observe the fact that the distance between two circles of latitude is always the same whereas the distance between two longitude meridians depends on the latitude angle, reaching its maximum when \( L = 0 \) and being zero when \( L = 90^\circ \).

In order to describe the orientation of the aircraft we need the three angles heading (\( \psi \)), pitch (\( \theta \)) and roll (\( \phi \)), depicted in figure 2.2. Heading is the angle between the aircrafts long axis projected onto the horizontal plane and a vector pointing to the north, where the long axis is the axis between the aircrafts tail and nose. Pitch is the angle between the aircrafts long axis and its projection onto the horizontal plane. Roll is the angle between the horizontal plane and the axis going through the wings. The pitch and roll are sometimes referred to as the aircrafts attitude.
The equations for latitude, longitude and altitude are

\[
\begin{align*}
\dot{L} &= \frac{v_n}{r_L + h}, \\
\dot{i} &= \frac{(r_i + h) \cos L}{v_e}, \\
\dot{h} &= -v_d.
\end{align*}
\]
In \((2.3)\), \(r_L\) and \(r_I\) are given by
\[
r_L = r_0 \frac{(1 - \varepsilon^2)}{(1 - \varepsilon^2 \sin^2 L)^{3/2}} \tag{2.4}
\]
\[
r_I = r_0 \frac{(1 - \varepsilon^2 \sin L)}{(1 - \varepsilon^2 \sin L)^{1/2}}
\]
where \(\varepsilon^2 = e(2 - e)\) according to Table 2.1. The velocity of the aircraft relative to the earth is given as the solution to the differential equation
\[
\dot{v}^n = C^n_b f^b - (\Omega^n_{en} + 2\Omega^n_{ie})v^n + g^n. \tag{2.5}
\]
In \((2.5)\), \(f^b\) is a specific force vector containing the accelerations sensed by the accelerometers. \(C^n_b\) is the transformation matrix from body frame to navigation frame. It depends on the attitude and heading of the aircraft according to
\[
C^n_b = \begin{pmatrix}
\cos \psi & -\sin \psi & 0 \\
\sin \psi & \cos \psi & 0 \\
0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
\cos \theta & 0 & \sin \theta \\
0 & 1 & 0 \\
-\sin \theta & 0 & \cos \theta
\end{pmatrix}
\begin{pmatrix}
1 & 0 & 0 \\
0 & \cos \phi & -\sin \phi \\
0 & \sin \phi & \cos \phi
\end{pmatrix}
\]
\[
\Omega = \begin{pmatrix}
0 & -\omega_z & \omega_y \\
\omega_z & 0 & -\omega_x \\
-\omega_y & \omega_x & 0
\end{pmatrix}
\]
\[
(2.6)
\]
The matrices \(\Omega^n_{en}\) and \(\Omega^n_{ie}\) describe the rotation of the navigation frame relative to the earth and the rotation of the earth relative to the inertial frame respectively. Both are expressed in navigation frame and in skew-symmetric form, i.e.
\[
\begin{pmatrix}
\cos L \\
0 \\
-\sin L
\end{pmatrix}
\omega^n_{ie}, \quad
\begin{pmatrix}
\frac{-v_x}{r_0 + h} \\
\frac{-v_y}{r_0 + h} \\
\frac{-v_z}{r_0 + h}
\end{pmatrix}
(2.7)
\]
\[
\omega^n_{en} = \begin{pmatrix}
\cos L \\
0 \\
-\sin L
\end{pmatrix}
\omega^n_{ie}, \quad
(2.8)
\]
Finally, \(g^n\) is given by
\[
g^n = \begin{pmatrix}
0 & 0 & g_d
\end{pmatrix}^T 
\]
\[
g_d = \frac{g_0 (1 + k_1 \sin^2 L)}{(1 - \varepsilon^2 \sin^2 L)^{1/2}} \left(1 - \frac{2h}{r_0}(1 + k_2 - 2e \sin^2 L) + \frac{3h^2}{r_0^2}\right). \tag{2.9}
\]
The corresponding matrix differential equation for \(C^n_b\) is given by
\[
\dot{C}^n_b = C^n_b \Omega^n_{eb} - \Omega^n_{en} C^n_b. \tag{2.10}
\]
The matrices \(\Omega^n_{eb}\) and \(\Omega^n_{en}\) in \((2.10)\) are given by \(\omega^n_{eb}\) and \(\omega^n_{en}\) according to \((2.7)\).
2.2 INS Error Dynamics

Due to the fact that the INS dynamics is very fast, it is easier to estimate the error in the INS, i.e.
\[ \dot{x} = \dot{x}^{\text{true}} - \dot{x}^{\text{ins}}, \] (2.11)
where \( \dot{x}^{\text{true}} \) are the true states and \( \dot{x}^{\text{ins}} \) are the states computed by the INS. This thesis will use the continuous state space model from [12] given by
\[ \dot{x}(t) = A(t)x(t) + B(t)u(t), \] (2.12)
where
\[ x = [\tilde{L} \ \tilde{l} \ \tilde{h} \ \tilde{v}_n \ \tilde{v}_e \ \gamma_n \ \gamma_e \ \gamma_d \ b_x^a \ b_y^a]^T, \] (2.13)
and \( u(t) \) is white noise with the components
\[ u = [u^h \ u^a_x \ u^a_y \ u^\gamma_x \ u^\gamma_y \ u^b_x \ u^b_y]^T. \] (2.14)
The matrices \( A(t) \) and \( B(t) \) are given in Appendix A. In sections 2.2.1 - 2.2.3 we will explain a bit about the different state variables but for detailed derivations please refer to [11] or [12].

2.2.1 Position Error Dynamics

Let us start with the position variables, \([\tilde{L} \ \tilde{l} \ h]^T\). Equations (2.3) and (2.11) give the following expressions for the latitude and longitude errors
\[ \dot{\tilde{L}} = \frac{v_n}{r_L + h} - \frac{v_n - \tilde{v}_n}{r_L - \tilde{r}_L + h - \tilde{h}}, \] \[ \dot{\tilde{l}} = \frac{v_e - \tilde{v}_e}{(r_l + h) \cos \tilde{L}} - \frac{v_e}{(r_l - \tilde{r}_l + h - \tilde{h}) \cos(L - \tilde{L})}. \] (2.15)
We will try to simplify these expressions. To do this we will analyze smaller parts of (2.15). If we apply Taylor expansion on (2.4) we obtain
\[ \frac{1}{r_L + h} = 1 - \frac{h}{r_0} + \frac{2 \epsilon^2 - 3 \epsilon^2 \sin^2 L}{2r_0} + O(\frac{\epsilon^4}{r_0}), \] \[ \frac{1}{r_l + h} = 1 - \frac{h}{r_0} - \frac{\epsilon^2 \sin^2 L}{2r_0} + O(\frac{\epsilon^4}{r_0}). \] (2.16)
This will give the basis of (2.15). In a similar way we can, together with the Taylor expansion
\[ \sin^2(L - \tilde{L}) = \sin^2 L + O(\tilde{L}), \] (2.17)
derive the following
\[ \frac{1}{r_L - \tilde{r}_L + h - \tilde{h}} = 1 - \frac{h}{r_0} + \frac{2 \epsilon^2 - 3 \epsilon^2 \sin^2 L}{2r_0} + O(\frac{\epsilon^4}{r_0}) + O(\frac{\tilde{h}}{r_0}), \] \[ \frac{1}{r_l - \tilde{r}_l + h - \tilde{h}} = 1 - \frac{h}{r_0} - \frac{\epsilon^2 \sin^2 L}{2r_0} + O(\frac{\epsilon^4}{r_0}) + O(\frac{\tilde{h}}{r_0}). \] (2.18)
The results in (2.16) and (2.18) together with the Taylor expansion
\[
\frac{1}{\cos(L - \tilde{L})} = \frac{1}{\cos L} = \frac{\sin L}{\cos^2 L} \tilde{L} + \mathcal{O}\left(\frac{\tilde{L}^2}{\cos^3 L}\right),
\] (2.19)
allows us to rewrite the equations for the latitude and longitude errors according to
\[
\dot{\tilde{L}} = \frac{\tilde{v}_n}{r_0} + \mathcal{O}\left(\frac{\tilde{v}_n \varepsilon^2}{r_0}\right) \approx \frac{\tilde{v}_n}{r_0},
\]
\[
\dot{\tilde{\ell}} = \frac{v_e \tilde{L} \sin L}{r_0 \cos^2 L} + \frac{\tilde{v}_e}{r_0 \cos L} + \mathcal{O}\left(\frac{\tilde{v}_e \varepsilon^2}{r_0 \cos L}\right) + \mathcal{O}\left(\frac{v_e \tilde{h} \varepsilon^2}{r_0 \cos L}\right),
\] (2.20)
The errors introduced by the approximations in (2.20) are small for the typical values involved and can be neglected [11]. The resulting equations seems rather intuitive. An error in the northern velocity component should result in an error in latitude. In the same way should an error in the eastern velocity component affect the longitude error. However, an error in the latitude should also affect the longitude error, due to the fact that the distance between two longitude meridians depends on the latitude.

If we look at the altitude error it can be seen from (2.3) that it simply becomes
\[
\dot{\tilde{h}} = -\tilde{v}_d,
\] (2.21)
i.e. it is linear from the start. In today’s aircraft the altitude channel in the INS is often stabilized with a barometric sensor. By measuring the air pressure this sensor determines the height above mean sea level. A barometric sensor alone is not accurate enough but combined with the data from the INS it delivers good altitude measurements. When using this type of stabilization, it is reasonable to assume that the pressure sensor drifts according to a random walk process, where the driving noise is described by \(u^h\). The INS altitude error will be assumed to follow this error which gives us
\[
\dot{\tilde{h}} \approx u^h.
\] (2.22)

### 2.2.2 Velocity And Attitude Error Dynamics

Now let us move on to the velocity variables, \([\tilde{v}_n, \tilde{v}_e]^T\). Our aim is to find a more simple expression for (2.5). This will be done by approximating the error in \(\Omega_{en}^n\) and \(\Omega_{ie}^n\) and by finding a simpler way to express the attitude error. Applying (2.11) on (2.5) gives us
\[
\dot{\tilde{v}}^n = C_b^n f^b + (C_b^n - \tilde{C}_b^n) \tilde{a}^b - (\tilde{\Omega}_{en}^n + 2\tilde{\Omega}_{ie}^n) \tilde{v}^n - (\tilde{\Omega}_{en}^n - \tilde{\Omega}_{en}^n + 2(\tilde{\Omega}_{ie}^n - \tilde{\Omega}_{ie}^n)) \tilde{v}^n + \tilde{g}^n,
\] (2.23)
with \(\tilde{a}^b = f^b - f_{b,\text{ins}}^b\) and \(\tilde{g}^n \approx (0 \ 0 \ \tilde{g}_d)^T\). Taylor expansion on \(\omega_{en}^n = \omega_{en}^n - \omega_{en,\text{ins}}^n\) using (2.8) and (2.11) around \(\varepsilon^2, \tilde{h}\) and \(L\) and on \(\omega_{ie}^n = \omega_{ie}^n - \omega_{ie,\text{ins}}^n\) around \(L\)
2.2 INS Error Dynamics

gives us
\[
\tilde{\omega}_{en}^n = \begin{bmatrix}
\frac{\nu_n}{r_1 + h} \\
- \frac{\nu_n}{r_1 + h} \\
\end{bmatrix} - \begin{bmatrix}
\frac{\nu_n - \nu_n}{r_1 - r_2 + h - h} \\
- \frac{\nu_n - \nu_n}{r_1 - r_2 + h - h} \\
\end{bmatrix}, \quad (2.24a)
\]
\[
\tilde{\omega}_{ie}^n = \omega_{ie} \begin{bmatrix}
\cos L \\
0 \\
- \sin L \\
\end{bmatrix} - \omega_{ie} \begin{bmatrix}
\cos(L - \tilde{L}) \\
0 \\
- \sin(L - \tilde{L}) \\
\end{bmatrix}. \quad (2.24b)
\]

If we consider earlier linearizations together with the Taylor expansions
\[
\tan(L - \tilde{L}) = \tan L - \frac{\tilde{L}}{\cos^2 L} + O(\tilde{L}^2) \quad (2.25a)
\]
\[
\sin(L - \tilde{L}) = \sin L - \tilde{L} \sin L + O(\tilde{L}^2) \quad (2.25b)
\]
\[
\cos(L - \tilde{L}) = \cos L + \tilde{L} \cos L + O(\tilde{L}^2) \quad (2.25c)
\]
applied on (2.24), we get
\[
\tilde{\omega}_{en}^n = \begin{bmatrix}
\frac{\nu_n}{r_0} \\
- \frac{\nu_n}{r_0} \\
\end{bmatrix} + \begin{bmatrix}
O\left(\frac{\nu_n^2}{r_0}\right) + O\left(\frac{\nu_n \kappa}{r_0}\right) \\
O\left(\frac{\nu_n^2}{r_0}\right) + O\left(\frac{\nu_n \kappa}{r_0}\right) \\
O\left(\frac{\nu_n \kappa}{r_0}\right) + O\left(\frac{\nu_n \kappa^2}{r_0}\right) \\
\end{bmatrix} \quad (2.26a)
\]
\[
\tilde{\omega}_{ie}^n = \omega_{ie} \begin{bmatrix}
- \tilde{L} \sin L \\
0 \\
- \tilde{L} \cos L \\
\end{bmatrix} + \omega_{ie} \begin{bmatrix}
\frac{\tilde{L}}{L} \\
0 \\
\frac{\tilde{L}}{L} \\
\end{bmatrix}. \quad (2.26b)
\]

Furthermore, simplifications of \(\omega_{en}^n - \tilde{\omega}_{en}^n\) and \(\omega_{ie}^n - \tilde{\omega}_{ie}^n\) in (2.23) can be done according to
\[
\omega_{en}^n - \tilde{\omega}_{en}^n = \begin{bmatrix}
\frac{\nu_n}{r_0} \\
- \frac{\nu_n}{r_0} \\
\end{bmatrix} + \begin{bmatrix}
\frac{\kappa}{L} + O\left(\frac{\nu_n \kappa}{r_0}\right) \\
\frac{\kappa}{L} + O\left(\frac{\nu_n \kappa}{r_0}\right) \\
O\left(\frac{\nu_n \kappa}{r_0}\right) + O\left(\frac{\nu_n \kappa^2}{r_0}\right) \\
\end{bmatrix} \quad (2.27a)
\]
\[
\omega_{ie}^n - \tilde{\omega}_{ie}^n = \omega_{ie} \begin{bmatrix}
\cos L \\
0 \\
- \sin L \\
\end{bmatrix} + \omega_{ie} \begin{bmatrix}
\frac{\tilde{L}}{L} \\
0 \\
\frac{\tilde{L}}{L} \\
\end{bmatrix}. \quad (2.27b)
\]

Both of the expressions (2.26) and (2.27) will be used to simplify (2.23).

Before we continue with the velocity variables we will look at the attitude error, \(\begin{bmatrix} \gamma_n & \gamma_e & \gamma_d \end{bmatrix}^T\). The matrix \(C^e_b\) describes the error in the rotation of the
computed navigation frame relative to the true navigation frame. $\tilde{C}_n$ is based upon combinations of sine and cosine operations performed on the error angles. These angles are in general small and this enables us to use the approximations $\sin \alpha \approx \alpha$ and $\cos \alpha \approx 1$. Let the vector $\gamma^n = [\gamma_n, \gamma_e, \gamma_d]^T$ form a skew-symmetric matrix, $\Gamma^n$, defined through

$$\tilde{C}_n = C^n - C_n^{ins} = C_n - C_n^{ins} C_b^n = (I - C_n^{ins}) C_b^n = \Gamma^n + O((\Gamma^n)^2).$$

Now we let $\gamma^n$ describe the attitude error instead of $\tilde{C}_n$. The gain with this approximation is that $\gamma^n$ only has three components that need to be updated, which is less than the nine components of $C_b^n$. The propagation equation for $\gamma^n$ is given by

$$\dot{\gamma}_n = C_b^n \dot{\omega}^b_{in} - \dot{\omega}^n_{in} - \Omega^n_{in} \gamma^n + \Delta \gamma^n,$$

where the error $\Delta \gamma^n$ is small [11].

Applying the approximations (2.26), (2.27) and (2.28) on (2.23) gives us

$$\dot{\tilde{v}}_n = -f^n \gamma^n + C^n f^b \dot{b} + \tilde{V}^n (\dot{\omega}^n_{en} + 2\omega^n_{ie}) + V^n (\dot{\omega}^n_{en} + 2\dot{\omega}^n_{ie}) + \tilde{g}^n + \Delta \tilde{v},$$

where we have used

$$f^n = C^n f^b,$$

$$\Gamma^n f^n = -F^n \gamma^n,$$

$$(\tilde{\Omega}^n_{en} + 2\omega^u_{ie}) \dot{b}^u = -\tilde{V}^n (\dot{\omega}^n_{en} + 2\omega^n_{ie}),$$

$$(\dot{\Omega}^n_{en} + 2\omega^u_{ie}) \dot{b}^u = -V^n (\dot{\omega}^n_{en} + 2\dot{\omega}^n_{ie}).$$

In Nordlund [11], it is shown that the error introduced in $\dot{\tilde{v}}_n$ and $\dot{\tilde{v}}_e$, here denoted by $\Delta \tilde{v}$, is small enough not to cause any significant error in the estimate.

### 2.2.3 Accelerometer Error Dynamics

The accelerometers and rate gyros suffer from an error characteristic that often can be rather complex. Titterton [14] gives an introduction to this. However, for this specific application accelerometer errors can be thought of as an offset plus white noise. Offsets for the rate gyros can be neglected [11]. The accelerometer offset can be modelled as

$$\dot{\tilde{b}}^a = -\frac{1}{\tau} \tilde{b}^a + u^b \approx u^b,$$

where the last approximation can be done due to the fact that $\tau$ generally is rather large. In this model we will only consider the accelerometer biases for the x- and y-directions in the body frame. This is because the altitude error already is compensated for by the stabilization of the vertical channel.
2.3 Measurement Model

In Chapter 1 we stated that terrain aided positioning (TAP) is based upon measurements of the terrain height below the aircraft. This section will conduct a more elaborate discussion regarding the measurement.

The measurement model for this application is

$$y_t = h\left(\begin{array}{c} L_t \\ l_t \end{array}\right) + e_t,$$  \hfill (2.33)

where $y_t$ is the measured terrain height, i.e. the difference between the altitude of the aircraft and the ground clearance measured by the radar altimeter. Furthermore, $h(\cdot)$ is the terrain height at position $[L_t\ l_t]^T$ given by the height database and $e_t$ is the measurement noise. In this model $e_t$ is a combination of the error induced by the radar measurement and the error induced by inaccuracies in the height database.

2.3.1 Radar Error

Look at Figure 2.3. To the left the aircraft measures the ground clearance, as it should. The error in this measurement could be modelled as a single Gaussian. To the right the beam from the radar hits the treetops and the measured ground clearance will be much smaller than the true clearance. Also this error can be modelled as a single Gaussian but with a different mean value and standard deviation than in the left case. Therefore the total error in the radar measurement can be modelled as a mixture of two Gaussians according to

$$p_{e_t} = \Pr(\lambda_1) \mathcal{N}(m_1, \sigma_1) + \Pr(\lambda_2) \mathcal{N}(m_2, \sigma_2),$$  \hfill (2.34)

where $\Pr(\lambda_1)$ and $\Pr(\lambda_2)$ are the probabilities that the radar altimeter measures to the ground and the treetops respectively.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure2_3.png}
\caption{Left: radar measurement to the ground. Right: radar measurement to the treetops.}
\end{figure}
2.3.2 Map Uncertainty

The height database used in this thesis has heights for positions in a grid where the grid points are separated in the horizontal plane. In order to obtain a height for a position between the grid points bilinear interpolation is used. Due to the non-linear characteristics of the height database the interpolation introduces an error in the measurement. This error can be modelled as additional uncertainty, i.e. additional covariance, in the measurement noise. The additional covariance will be denoted by $R_{\text{add}}$. The measurement model now becomes

$$p_e = \Pr(\lambda_1)N(m_1, \sigma_1 + R_{\text{add}}) + \Pr(\lambda_2)N(m_2, \sigma_2 + R_{\text{add}}). \quad (2.35)$$

2.4 Summary Of Chapter

In this chapter we presented the state space model given by equations (2.12)-(2.14). Some areas of the state space model were highlighted. Also featured was the measurement model given by (2.33) and a discussion regarding the measurement errors.
Chapter 3

State Estimation
Background Theory

This chapter presents the Kalman filter and the GPB1 filter – linear filtering techniques that will be used later on in the sensor fusion. It also describes the principles behind non-linear Bayesian estimation theory and non-linear filter techniques such as the particle filter and the marginalized particle filter. These filtering techniques form the basis of the marginalized particle filter. This filter is the one that will be applied to the state space model from Chapter 2. At the end of this chapter a method of evaluating a non-linear filter, such as the marginalized particle filter, called the Cramér-Rao lower bound is described.

The introduction to the particle filter and the marginalized particle filter are only descriptive, for more details see for instance Nordlund [11] or Karlsson [10].

3.1 Kalman Filter

Consider the linear state space model

\[ x_{t+1} = F_t x_t + G_t u_t, \]
\[ y_t = H_t x_t + e_t. \]  

(3.1)

In (3.1) \( x_t \) represents the true states of the system and \( y_t \) is the measured quantity. Furthermore, \( u_t \) and \( e_t \) are process noise and measurement noise respectively. They are independent and Gaussian distributed, i.e.

\[ u_t \sim \mathcal{N}(0, Q_t), \quad e_t \sim \mathcal{N}(0, R_t). \]  

(3.2)

In filtering applications the aim is to get an as accurate estimation as possible of the true states, \( x_t \), from the noisy measurements, \( y_t \). For a linear model, such as the one in (3.1), the optimal filter for this task is the well known Kalman filter (KF). When using the KF the estimate, \( \hat{x}_{t|t} \), and its covariance, \( P_{t|t} \), propagate
\[ \hat{x}_{t|t} = \hat{x}_{t|t-1} + P_{t|t-1}H_t^TS_t^{-1}(y_t - H_t\hat{x}_{t|t-1}), \]
\[ P_{t|t} = P_{t|t-1} - P_{t|t-1}H_t^TS_t^{-1}H_tP_{t|t-1}, \]
\[ S_t = R_t + H_tP_{t|t-1}H_t^T, \]
\[ \hat{x}_{t+1|t} = F_t\hat{x}_{t|t}, \]
\[ P_{t+1|t} = F_tP_{t|t}F_t^T + G_tQ_tG_t^T, \]

(3.3)

with the initial values \( \hat{x}_{0|-1} = 0 \) and \( P_{0|-1} = P_0 \). The KF will not be investigated further here but for a more elaborate introduction and a formal proof, please refer to Gustafsson [7].

3.2 Multiple Models

As stated before, the KF is optimal for the linear and Gaussian case but when the noise is non-Gaussian, the KF is only the optimal filter among all linear filters. When the noise is a sum of Gaussians, multiple models can be used. The purpose of multiple models is to provide a filter that combines several Kalman filters. One such multiple model is the generalized pseudo-Bayesian (gpb). For more on these type of filters see Gustafsson [6]. The model we consider is

\[ x_{t+1} = F_t(\lambda_{t+1})x_t + G_t(\lambda_{t+1})u_t, \]
\[ y_t = H_t(\lambda_t)x_t + e_t(\lambda_t), \]

(3.4)

where \( \lambda_t \in \{1, 2, \ldots, M\} \) is a discrete random variable describing what mode the system is in and

\[ e_t(\lambda_t) = \begin{cases} 
N(m_1, \sigma_1), & \lambda_t = 1 \\
N(m_2, \sigma_2), & \lambda_t = 2 \\
\vdots & \\
N(m_M, \sigma_M), & \lambda_t = M 
\end{cases} \]

(3.5)

In order to get a better understanding of the mode variable, \( \lambda_t \), let us look at what \( \lambda_t \) means in this thesis. From section 2.3.1 we learned that the beam from the radar altimeter either hits the ground or the treetops. These two cases can be considered as two different modes, \( \lambda_1 \) and \( \lambda_2 \). This means that \( e_t(\lambda_t) \) will have the same form as (2.35), i.e. it consists of two Gaussians. Therefore, the number of possible modes will be \( M = 2 \) in the sequel.

If the mode sequence, \( \Lambda_t = \{\lambda_0, \ldots, \lambda_t\} \), is known we are back at the model according to (3.1). The law of total probability gives us the filtering posterior probability density as

\[ p(x_t|Y_t) = \sum_{\Lambda_t} p(x_t|Y_t, \Lambda_t)p(\Lambda_t|Y_t), \]

(3.6)

where \( Y_t = \{y_0, \ldots, y_t\} \). Unfortunately it is impossible to use the Kalman filter to optimally estimate the posterior density. At time \( t \) there will be \( 2^{t+1} \) possible mode sequences to account for. One solution is to merge hypotheses older than
some threshold, which makes the computational load constant over time. To only keep hypotheses from time \(t - L + 1\) means that only \(2^L\) different hypotheses need to be considered. This means \(2^L\) Kalman filters at each time step. This is what the GPBL does, where \(L\) denotes the order of the GPB. The new filtering posterior probability density now becomes

\[
p(x_t|Y_t) \approx \sum_{\Lambda_t-L+1} p(x_t|Y_t, \Lambda_t)p(\Lambda_t|Y_{t-1}) (3.7)
\]

In this thesis we will have an order \(L = 1\). The GPB1 now runs \(2^1 = 2\) Kalman filters in parallel, where the first filter runs under the assumption that the beam from the radar altimeter hit the ground and the second filter assumes that it hit the treetops. One drawback with using the order \(L = 1\) is the fact that the probability computed for each mode at time \(t\) will not depend on earlier probabilities. It is quite intuitive to imagine that if the beam from the radar altimeter hit the ground the last \(L > 1\) times, chances are good that it will do that at time \(t\) as well. This could be the case when flying over an open field. By taking earlier probabilities into account, i.e. let the order \(L > 1\), the complexity of the GPBL filter will increase too much. That is why \(L = 1\) in this thesis.

Next the GPB1 tries to estimate the probability of each mode and then merge the two Kalman filters according to their probabilities, \(p(\lambda_t|Y_{t-1}) = p(\lambda_t)\). That result combined with using the well-known Baye’s rule

\[
p(x_t|Y_t) = \frac{p(Y_t|X_t)p(x_t)}{p(Y_t)} (3.8)
\]

repeatedly gives us an expression for \(p(\lambda_t|Y_t)\) as

\[
p(\lambda_t|Y_t) = \frac{p(y_t|\lambda_t, Y_{t-1})p(\lambda_t|Y_{t-1})}{p(y_t|Y_{t-1})} = \frac{p(y_t|\lambda_t, Y_{t-1})p(\lambda_t)}{p(y_t|Y_{t-1})} \propto p(y_t|\lambda_t, Y_{t-1})p(\lambda_t). (3.9)
\]

The last step in (3.9) can be done due to the fact that the denominator will be the same for all modes. It will therefore be computationally more tractable to calculate the numerator and normalize

\[
\alpha_t^{(i)} = p(y_t|\lambda_t = i, Y_{t-1})p(\lambda_t = i) (3.10a)
\]

\[
\tilde{\alpha}_t^{(i)} = \frac{\alpha_t^{(i)}}{\sum_j \alpha_t^{(j)}}, (3.10b)
\]

where \(i\) denotes the mode.

A short summary of the GPB1 filter is given in Algorithm 3.1.

**Algorithm 3.1** The GPB1 filter with two modes
1. Consider the two possible modes at time $t$.

2. Apply a Kalman filter for each mode $i$, giving $\hat{x}_t|_{t-1}$, $\hat{x}^{(i)}_t$, $P_t|_{t-1}$ and $P^{(i)}_t$.

3. Compute the weights for each mode, $\bar{\alpha}^{(i)}_t$, from
\[ p(y_t|\lambda_t = i, Y_{t-1}) = \mathcal{N}(y_t - H_t\hat{x}_t|_{t-1} - m_i, \sigma_i + H_t P_{t|t-1} H^T_t). \]

4. Merge the two modes, giving
\[ \hat{x}_t = \sum_{i=1}^{2} \bar{\alpha}^{(i)}_t \hat{x}^{(i)}_t \]
\[ P_t = \sum_{i=1}^{2} \bar{\alpha}^{(i)}_t \left( P^{(i)}_t + \left( \hat{x}^{(i)}_t - \hat{x}_t \right) \left( \hat{x}^{(i)}_t - \hat{x}_t \right)^T \right). \]

### 3.3 Recursive Bayesian Estimation

Bayesian estimation theory considers everything unknown as a stochastic variable. This leads to a description where the initial or prior distribution is assumed to be known. By using observations the estimate can then later be revised by computing the posterior density. First, consider a state space model where noise enters additively, i.e.

\[ x_{t+1} = f(x_t) + G_t u_t, \quad (3.11a) \]
\[ y_t = h(x_t) + e_t. \quad (3.11b) \]

As before, $x_t$ represents the true, unknown states and $y_t$ is the measured quantity. $u_t$ and $e_t$ are process noise and measurement noise with probability densities $p(u_t)$ and $p(e_t)$ respectively. Let $X_t = \{x_0, \ldots, x_t\}$ and $Y_t = \{y_0, \ldots, y_t\}$ be the stacked vectors of all the states and measurements up to time $t$. With assumptions that $u_t$ and $e_t$ are independent of each other it can be shown that $x_t$ is a Markov process

\[ p(X_t) = \prod_{k=0}^{t} p(x_k|x_{k-1}), \quad (3.12) \]

and the measurements, $Y_t$, are conditionally given by the states, $X_t$,

\[ p(Y_t|X_t) = \prod_{k=0}^{t} p(y_k|x_k). \quad (3.13) \]

These expressions grow as time goes on. A desirable solution would be to have recursive expressions. From Baye’s formula \((3.8)\) and \((3.12)-(3.13)\) the following
3.4 Particle Filter

Particle filters, or Monte Carlo methods, have been a growing research field lately due to improved computer performance. They provide solutions to many problems, where linearizations and Gaussian approximations are intractable or would yield too low performance. The particle filter creates a large number of possible states, using the propagation equations from (3.11). These states form a cloud of particles, hence the name particle filters, and is used to obtain an empirical approximation of the joint posterior distribution. The particle filter consists of two basic parts: importance sampling and resampling.

3.4.1 Importance Sampling

In the process of importance sampling a number of samples are drawn, so many that they represent the joint posterior distribution, $p(X_t|Y_t)$. Now, it would be desirable to draw samples directly from $p(X_t|Y_t)$ but since the true states are unknown this is not possible. And if we already knew the true states, filtering would not be necessary at all. Instead we draw $N$ independent realizations of $X_t$ from the known probability density $q(X_t|Y_t)$. The function $q(X_t|Y_t)$ is often referred to as the importance function. This function should be easy to draw samples from and the samples are denoted by

$$\left\{ X^{(i)}_t \right\}_{i=1}^N = \left\{ x^{(i)}_0, \ldots, x^{(i)}_t \right\}_{i=1}^N .$$

Our best hope is that $q(X_t|Y_t)$ is as close to $p(X_t|Y_t)$ as possible. Let

$$w^{(i)}_t \propto \frac{p(X^{(i)}_t|Y_t)}{q(X^{(i)}_t|Y_t)}$$

define the importance weight for each sample. This can be thought upon as how well $q(X_t|Y_t)$ approximates $p(X_t|Y_t)$. Of course the support of $p(X_t|Y_t)$ must be included in $q(X_t|Y_t)$. By using Baye’s rule, the definition of conditional probability
and the Markov property introduced in (3.12) and 3.13 we obtain
\[
p(X_t|Y_t) = \frac{p(y_t|X_t, Y_{t-1})p(X_t|Y_{t-1})}{p(y_t|Y_{t-1})} \\
= \frac{p(y_t|x_t)p(x_t|X_{t-1}, Y_{t-1})p(X_{t-1}|Y_{t-1})}{p(y_t|Y_{t-1})} \\
= \frac{p(y_t|x_t)p(x_t|x_{t-1})}{p(y_t|Y_{t-1})}p(X_{t-1}|Y_{t-1}).
\] (3.17)

By ignoring the normalization factor we get
\[
p(X_t|Y_t) \propto p(y_t|x_t)p(x_t|x_{t-1})p(X_{t-1}|Y_{t-1}).
\] (3.18)

Next, we assume that the importance function is chosen such that
\[
q(X_t|Y_t) = q(x_t|X_{t-1}, Y_t)q(X_{t-1}|Y_{t-1}).
\] (3.19)

By inserting (3.18) and (3.19) into (3.16) we get a recursive formula for updating
the weights as
\[
w_t^{(i)} \propto \frac{p(y_t|x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)})p(X_{t-1}^{(i)}|Y_{t-1})}{q(x_t^{(i)}|X_{t-1}^{(i)}, Y_t)q(X_{t-1}^{(i)}|Y_{t-1})} = \frac{p(y_t|x_t^{(i)})p(x_t^{(i)}|x_{t-1}^{(i)})}{q(x_t^{(i)}|X_{t-1}^{(i)}, Y_t)}w_{t-1}^{(i)}. \tag{3.20}
\]

Now a simple choice of \( q(x_t^{(i)}|X_{t-1}^{(i)}, Y_t) = p(x_t^{(i)}|x_{t-1}^{(i)} \) gives us
\[
w_t^{(i)} \propto p(y_t|x_t^{(i)})w_{t-1}^{(i)} = p_{e_t}(y_t - h(x_t^{(i)}))w_{t-1}^{(i)}, \tag{3.21}
\]
and the normalized weights simply become
\[
\tilde{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_j w_t^{(j)}}. \tag{3.22}
\]

### 3.4.2 Resampling

Doucet et al. [4] pointed out that the variance of the importance weights can only increase over time. This means that fewer of the samples contributes to the posterior density, their weights tend to zero, and the estimated density does not reflect the true one. This is illustrated in Figure 3.1. The solution to overcome this problem is resampling. The idea with resampling is to draw a new set of samples, \( \{x_t^{(i)}\}_{i=1}^N \), from the samples we already have, \( \{x_t^{(i)}\}_{i=1}^N \). This should be done so that the probability to draw sample \( i \) should be equal to the normalized weight for that sample, i.e. \( \Pr(x_t^{(k)} = x_t^{(i)}) = \tilde{w}_t^{(i)} \). Samples with weights close to zero are discarded while samples with higher weights are duplicated, by doing this. In [8] four different resampling techniques are compared with respect to performance and complexity.

A resampling could be performed at every iteration in the particle filter, however this may not be the optimal solution. Better is to let a predefined criterion
such as the effective sample size, $N_{eff}$, decide when to perform a resampling operation. This method calculates the number of particles in the cloud that actually contributes to the probability density approximation. Analytically it is impossible to evaluate $N_{eff}$ but a good approximation is given by

$$\hat{N}_{eff} \approx \frac{1}{\sum_{i=1}^{N} (w_i^{(i)})^2}.$$ (3.23)

Now a resampling should be performed every time the effective sample size is less than some predefined threshold, $N_{th}$. After resampling, all the weights should be set to $1/N$.

When resampling is performed another problem occurs; the samples become dependent of each other. One way to overcome this problem is to let the model exhibit more process noise than the true system really does. This is done by adding artificial noise, $u_{add}^{(i)}$, in the state propagation equations. A common choice for the distribution of $u_{add}^{(i)}$ is

$$u_{add}^{(i)} \sim \mathcal{N}(0, \kappa P_t).$$ (3.24)

$P_t$ is the estimated covariance of $x_t$ and $\kappa$ is a factor typically in the order of 0.01-0.001.

In Algorithm 3.2 a simple summary of the particle filter is given.

**Algorithm 3.2** The particle filter algorithm [4].
1. Set \( t = 0 \), generate \( N \) samples from \( \{x_0^{(i)}\}_{i=1}^N \) from an initial distribution. Initialize the importance weights \( \bar{w}_i^{(0)} = 1/N, \quad i = 1, \ldots, N \).

2. Compute the importance weights \( w_i^{(t)} = p(y_t|x_t^{(i)})\bar{w}_i^{(t-1)} \) and normalize, i.e., \( \bar{w}_i^{(t)} = w_i^{(t)} / \sum_j \bar{w}_j^{(t)} \).

3. If resampling should be performed, generate a new set \( \{x_t^{(k)}\}_{k=1}^N \) by resampling \( N \) times from \( \{x_t^{(i)}\}_{i=1}^N \), with \( \text{Pr}(x_t^{(k)} = x_t^{(i)}) = \bar{w}_i^{(t)} \) and set \( \bar{w}_i^{(t)} = 1/N \).

4. Predict new particles, \( x_{t+1}^{(i)} = f(x_t^{(k)}) + G_t u_t + u_{t}^{ad} \).

5. Increase \( t \) and continue from step 2.

### 3.5 Marginalized Particle Filter

The number of particles needed for a good approximation of the posterior density probability is increasing with the dimension of the estimated state vector \( x_t \). Therefore, whenever a linear-Gaussian sub-structure exists, this can be used to reduce the number of particles. This is done by marginalizing out the linear part of the pdf \( p(x_t|Y_t) \). Denote the linear part with \( x_t^k \) and the non-linear part with \( x_t^p \). Now we have \( X_t^k = \{x_t^k\}_{i=0}^T \) and \( X_t = \{x_t^k, X_t^p\} \). This gives us

\[
p(X_t|Y_t) = p(x_t^k, X_t^p|Y_t) = p(x_t^k|X_t^p, Y_t)p(X_t^p|Y_t),
\]

where \( p(x_t^k|X_t^p, Y_t) \) is given by a relatively fast Kalman filter and \( p(X_t^p|Y_t) \) is given by a particle filter. This type of filter is called a Rao-Blackwellized filter or a marginalized particle filter (MPF). Consider the state space model (2.12)-(2.14), which was presented in Chapter 2. In this model, only the variables that describes position are highly non-linear. All other variables can be linearized without significant losses. Hence, when we use the marginalized particle filter we separate the position variables from the others, which gives us the continuous state space model

\[
x_t = [(x_t^p)^T \quad (x_t^k)^T]^T,
\]

where

\[
x_t^p = [\tilde{L}_t \quad \tilde{l}_t \quad \tilde{h}_t]^T,
\]

\[
x_t^k = [\tilde{v}_{n,t} \quad \tilde{v}_{e,t} \quad \gamma_{n,t} \quad \gamma_{e,t} \quad \gamma_{d,t} \quad b_y^a \quad b_y^b]^T.
\]

The superscripts \( p \) and \( k \) indicate that the variables should be estimated with the particle filter and the Kalman filter respectively.
3.5 Marginalized Particle Filter

The discrete state space model now becomes

\[
\begin{bmatrix}
  x_{t+1}^p \\
  x_{t+1}^k
\end{bmatrix} = F_t \begin{bmatrix}
  x_t^p \\
  x_t^k
\end{bmatrix} + G_t \begin{bmatrix}
  u_t^h \\
  u_t^k
\end{bmatrix},
\]

(3.28a)

\[
y_t = h \left( \begin{bmatrix} L_{t}^{\text{ins}} \\ x_{t,1}^p \end{bmatrix} + \begin{bmatrix} x_{t,1}^p \\ x_{t,2}^p \end{bmatrix} \right) - x_{t,3}^k + e_t,
\]

(3.28b)

where

\[
F_t = I + T_s A_t = \begin{bmatrix}
  F_{p,t}^p & 0 \\
  0 & F_{k,t}^k
\end{bmatrix},
\]

\[
G_t = T_s (I + \frac{T_s}{2} A_t) B_t = \begin{bmatrix}
  0 & G_t^p \\
  T_s(1 + \frac{T_s}{2}) G_t^k \\
  0 & 0
\end{bmatrix},
\]

\[
u_t = [u_t^h, u_t^k]^T, \quad u_t^k = [u_{x,t}^g, u_{y,t}^g, u_{x,t}^g, u_{y,t}^g, u_{x,t}^b, u_{y,t}^b]^T.
\]

As discussed in section (3.4.2) some artificial noise will also be added which gives us

\[
x_{t+1}^p = F_{p,t}^p x_t^p + F_{k,t}^p u_t^k + u_{t,\text{add}}^p.
\]

(3.30)

A summary of the algorithm is given in Algorithm 3.3.

**Algorithm 3.3** The Ordinary MPF

1. Initialization:
   For \( i = 1, \ldots, N \), sample \( x_0^{n,(i)} \sim p(x_0^n) \), and set
   \[
   \begin{align*}
   \{x_{0,-1}^{k,(i)}, P_{0,-1}^k\} & = \{0, P_0^k\}, \\
   \{x_{0,-1}^{g,(i)}, P_{0,-1}^g\} & = \{0, P_0^g\} \\
   \bar{w}_{-1}^{(i)} & = 1/N.
   \end{align*}
   \]

2. Particle filter measurement update:
   For each \( i = 1, \ldots, N \), update
   \[
   w_t^{(i)} = p_{e_t} \left( y_t - h \left( \begin{bmatrix} L_{t}^{\text{ins}} \\ x_{t,1}^p \end{bmatrix} + \begin{bmatrix} x_{t,1}^{p,(i)} \\ x_{t,2}^{p,(i)} \end{bmatrix} \right) \right) \bar{w}_{t-1}^{(i)},
   \]
   \[
   \bar{w}_{t}^{(i)} = w_t^{(i)} / \sum_{i} w_t^{(i)}.
   \]

3. Resampling:
   If
   \[
   \hat{N}_{\text{eff}} \approx \frac{1}{\sum_{i=1}^{N} (w_t^{(i)})^2} < N_{\text{th}},
   \]
   resample \( N \) times with replacement according to
   \[
   \Pr(x_t^{(i)} = x_t^{(k)}) = \bar{w}_{t}^{(i)}.
   \]
4. Kalman filter measurement update:
For each \(i = 1, \ldots, N\), set
\[
\hat{x}_{t|t}^{k,(i)} = \hat{x}_{t|t-1}^{k,(i)}, \quad P_{t|t}^{k} = P_{t|t-1}^{k}.
\]

5. Particle filter time update:
For \(i = 1, \ldots, N\), sample
\[
x_{t+1}^{p,(i)} \sim N(\hat{x}_{t+1|t}^{p}, P_{t+1|t}^{p}),
\]
where
\[
\begin{align*}
\hat{x}_{t+1|t}^{p} &= f_{t}^{p}(x_{t+1}^{p}) + F_{t}^{p}\hat{x}_{t|t}^{k}, \\
P_{t+1|t}^{p} &= F_{t}^{p}P_{t|t}^{k}(F_{t}^{p})^{T} + Q_{t}^{add}.
\end{align*}
\]

6. Kalman filter time update:
For each \(i = 1, \ldots, N\), compute
\[
\begin{align*}
\hat{x}_{t+1|t}^{k} &= (F_{t}^{k} - K_{p,t}F_{t}^{p})\hat{x}_{t|t}^{k} + \\
&\quad K_{p,t}(x_{t+1}^{p} - f_{t}^{p}(x_{t}^{p})) + f_{t}^{k}(x_{t}^{k}) \\
P_{t+1|t}^{k} &= F_{t}^{k}P_{t|t}^{k}(F_{t}^{k})^{T} + G_{t}^{k}Q_{t}^{k}(G_{t}^{k})^{T} - \\
&\quad K_{p,t}S_{p,t}K_{p,t}^{T}.
\end{align*}
\]
where
\[
\begin{align*}
K_{p,t} &= F_{t}^{k}P_{t|t}^{k}(F_{t}^{p})^{T}S_{p,t}^{-1}, \\
S_{p,t} &= Q_{t}^{add} + F_{t}^{p}P_{t|t}^{k}(F_{t}^{p})^{T}.
\end{align*}
\]

This algorithm will be compared to an extended version of the MPF and will therefore in the sequel be denoted as an ordinary MPF.

3.6 Extending The Marginalized Particle Filter

The altitude channel can not be linearized without significant losses. Thanks to multiple models, discussed in section 3.2, we can use the GPB1 filter to estimate the altitude instead. By doing this we only have two variables that need to be estimated by the particle filter, and the total number of particles needed for a good estimation decreases. When we use the extended marginalized particle filter we separate the horizontal position and altitude variables from the others
\[
x_{t} = [(x_{t}^{p})^{T} (x_{t}^{g})^{T} (x_{t}^{k})^{T}]^{T},
\]
where
\[
\begin{align*}
x_{t}^{p} &= [\tilde{L}_{t} \tilde{L}_{t}]^{T}, \\
x_{t}^{g} &= \tilde{h}_{t}, \\
x_{t}^{k} &= [\tilde{v}_{n,t} \tilde{v}_{e,t} \gamma_{n,t} \gamma_{e,t} \gamma_{d,t} b_{n,t}^{0} b_{e,t}^{0}]^{T}.
\end{align*}
\]
The superscripts \( p, q \) and \( k \) indicate that the variables should be estimated with the particle filter, the GPB1 filter and the Kalman filter respectively.

The discrete state space model now becomes

\[
\begin{bmatrix}
    x_{t+1}^p \\
    x_{t+1}^q \\
    x_{t+1}^k
\end{bmatrix} = \begin{bmatrix}
    F_t & 0 & 0 \\
    0 & F_t & 0 \\
    0 & 0 & F_t
\end{bmatrix}
\begin{bmatrix}
    x_t^p \\
    x_t^q \\
    x_t^k
\end{bmatrix} + \begin{bmatrix}
    u_t^p \\
    u_t^q \\
    u_t^k
\end{bmatrix},
\]

where

\[
F_t = I + T_s A_t = \begin{bmatrix}
    F_{p,t}^p & 0 & F_{k,t}^p \\
    0 & 1 & 0 \\
    0 & 0 & F_{k,t}^k
\end{bmatrix},
\]

\[
G_t = T_s(I + \frac{T_s}{2} A_t)B_t = \begin{bmatrix}
    0 & G_t^p \\
    T_s(1 + \frac{T_s}{2}) & 0 \\
    0 & G_t^k
\end{bmatrix},
\]

\[
u_t = [u_t^p, u_t^q, u_t^k]^T, \quad u_t^k = [u_{s,t}^q, u_{g,t}^q, u_{a,t}^q, u_{b,t}^q, u_{x,t}^q, u_{y,t}^q, u_{z,t}^q, u_{x,t}^b, u_{y,t}^b, u_{z,t}^b]^T.
\]

Again, some artificial noise will be added, which gives us

\[
x_{t+1}^p = F_p x_t^p + F_p u_t^k + u_t^{add}.
\]

A summary of the applied algorithm is given in Algorithm 3.4.

**Algorithm 3.4** The MPF for blended INS/TAP [12]

1. **Initialization:**
   For \( i = 1, \ldots, N \), sample \( x_0^{n,i} \sim p(x_0^n) \), and set
   \[
   \begin{align*}
   \{ & x_{0|0}^{k,i}, p_{0|0}^k, p_{0|0}^q, p_{0|0}^p, \alpha_{-1} \} = \{0, P_0^k, P_0^q, P_0^p, 0\}, \\
   \{ & \sigma_{-1}^{(1), i}, \sigma_{-1}^{(2), i} \} = \{\text{Pr}(\lambda_t = 1), \text{Pr}(\lambda_t = 2)\}.
   \end{align*}
   \]

2. **GPB1 filter measurement update:**
   For \( i = 1, \ldots, N \) and \( j_t = 1, 2 \), compute
   \[
   \begin{align*}
   x_{t|t-1}^{g,i,j_t} &= \hat{x}_{t|t-1}^{g,i,j_t} - P_{t|t-1}^{g,i,j_t} (F_{f,t}^{g,i,j_t})^{-1} (y_t - \hat{y}_{t|t-1}^{(i,j_t)}), \\
   p_{t|t}^{g,i,j_t} &= p_{t|t-1}^{g,i,j_t} - P_{t|t-1}^{g,i,j_t} (F_{f,t}^{g,i,j_t})^{-1} (p_{t|t-1}^{g,i,j_t})^T, \\
   \alpha_{t}^{i,j_t} &= \mathcal{N}(\hat{y}_{t|t-1}^{(i,j_t)}, S_{f,t}^{g,i,j_t}) \sum_{j_{t-1}=1}^{2} \sigma_{j_{t-1}}^{i,j_{t-1}}.
   \end{align*}
   \]

   where
   \[
   \begin{align*}
   \hat{y}_{t|t-1}^{(i,j_t)} &= h(x_t^{p,i}) + \hat{x}_{t|t-1}^{g,i,j_t} - m_{t}^{j_t}, \\
   S_{f,t}^{g,i,j_t} &= R_f^{j_t} + p_{t|t-1}^{g,i,j_t}.
   \end{align*}
   \]
Then compute
\[ \bar{\alpha}_t^{(i,j_t)} = \frac{\alpha_t^{(i,j_t)}}{\alpha_t^{(i,1)} + \alpha_t^{(i,2)}}, \]
\[ \hat{x}_t^{g,(i)} = \sum_{j_{t-1}=1}^{2} \alpha_t^{(i,j_t)} \hat{x}_t^{g,(i,j_t)}, \]
\[ P_t^{d,(i)} = \sum_{j_{t-1}=1}^{2} \alpha_t^{(i,j_t)} (P_t^{d,(i,j_t)} + (\hat{x}_t^{g,(i,j_t)} - \hat{x}_t^{g,(i)}))^2). \]

3. Particle filter measurement update:
For each \( i = 1, \ldots, N \), update
\[ w_t^{(i)} = \bar{w}_{t-1}^{(i)} \sum_{j_{t}=1}^{2} \alpha_t^{(i,j_t)}, \]
\[ \bar{w}_t^{(i)} = w_t^{(i)} / \sum_{i=1}^{N} w_t^{(i)}. \]

4. Resampling:
If
\[ \hat{N}_{\text{eff}} \approx \frac{1}{\sum_{i=1}^{N} (\bar{w}_t^{(i)})^2} < N_{th}, \]
resample \( N \) times with replacement according to
\[ Pr(x_t^{(i)} = x_t^{(k)}) = \bar{w}_t^{(i)}. \]

5. Kalman filter measurement update:
For each \( i = 1, \ldots, N \), set
\[ \hat{x}_t^{k,(i)} = \hat{x}_t^{k,(i)} \]
\[ P_t^{k} = P_t^{k}. \]

6. GPB1 filter time update:
For each \( i = 1, \ldots, N \), compute
\[ \hat{x}_{t+1|t}^{g,(i)} = x_{t|t}^{g,(i)} \]
\[ P_{t+1|t}^{g,(i)} = P_{t|t}^{g,(i)} + T_s (1 + T_s / 2)^2 Q_t^{g}. \]

7. Particle filter time update:
For \( i = 1, \ldots, N \), sample
\[ x_{t+1|t}^{p,(i)} \sim \mathcal{N}(\hat{x}_{t+1|t}^{p}, P_{t+1|t}^{p}), \]
where
\[ \hat{x}_{t+1|t}^{p} = P_t^{p} (x_t^{p}) + \hat{x}_{t|t}^{k}, \]
\[ P_{t+1|t}^{p} = P_t^{p} P_t^{p} (P_t^{p})^T + Q_t^{add}. \]
8. Kalman filter time update:
For each \( i = 1, ..., N \), compute

\[
\hat{x}_{t+1|t}^k = (F_t^k - K_{p,t} F_p^k) \hat{x}_{t|t}^k + \\
K_{p,t} (x_{t+1}^p - f_t^p (x_t^p)) + f_t^p (x_t^p)
\]

\[
P_{t+1|t}^k = F_t^k P_t^k (F_t^k)^T + G_t^k Q_t^k (G_t^k)^T - \\
K_{p,t} S_{p,t} K_{p,t}^T,
\]

where

\[
K_{p,t} = F_t^k P_t^k (F_t^k)^T S_{p,t}^{-1},
\]

\[
S_{p,t} = Q_{add} + F_t^k P_t^k (F_t^k)^T.
\]

This is the algorithm that will be evaluated in this thesis. When no confusion can arise, this will simply be denoted the marginalized particle filter (MPF).

As discussed in section 3.2, the GPBF filter used in this thesis does not depend on the mode history. This may cause some loss of information and introduce an error in the altitude estimation, although the error should be small [12]. A better estimation of the altitude could lead to a better estimation of the horizontal position as well. This is due to the fact that the estimate of the ground elevation strongly depends on a good altitude estimation. When the altitude variable is estimated in the particle filter as in the ordinary MPF, there should be no losses of this kind, although the number of particles needed increases. This thesis will compare the MPF presented in this section with the ordinary MPF. The purpose of this comparison is to see what the performance gain is when using the ordinary MPF – if any – and put that in relation to the computational costs. Whenever comparisons are made with the ordinary MPF, the MPF in this section will be called the extended MPF.

### 3.7 Cramér-Rao Lower Bound

For non-linear filtering applications the optimal solution is generally intractable. But the performance of the optimal solution can often be computed with limited resources. A lower bound for the optimal estimation covariance can be used as to evaluate how well the implemented solution performs. One such lower bound is the Cramér-Rao Lower Bound (CRLB). Bergman [11] explains the CRLB thoroughly.

According to Nordlund [12] the CRLB for a system such as the one considered in this thesis will be

\[
P_{0_{t+1}}^{CR} = P_0,
\]

\[
P_{t+1}^{CR} = F_t P_{t+1}^{CR} (I - (I + R_t^{-1} P_t^{CR})^{-1} R_t^{-1} P_t^{CR}) F_t^T + G_t Q_t G_t^T,
\]

(3.36)
where $R^{-1}$ is given by

$$R^{-1} = E_{p(e_t)} \left[ \left( \frac{\partial}{\partial e_t} \log p(e_t) \right)^2 \right] \times E_{p(x^n_t)} \left[ \begin{array}{c} \frac{\partial}{\partial x^n_t} h(x^n_t) \\ -1 \\ 0_{7\times1} \end{array} \right] \left[ \begin{array}{c} \frac{\partial}{\partial x^n_t} h(x^n_t) \\ -1 \\ 0_{7\times1} \end{array} \right]^T \right].$$

(3.37)

### 3.8 Summary Of Chapter 3

In this chapter it was shown how the Kalman filter, the GPB1 filter and the particle filter form the marginalized particle filter presented in section 3.6. This filtering technique was then applied on the state space model presented in Chapter 2. In the next chapter the performance of this filter will be tested on real flight data.
Chapter 4

Performance Evaluation On Test Flights

This chapter discusses the design parameters in the MPF presented in section 3.6. The MPF is applied on the state space model from Chapter 2 and tested on real flight data. The flights and the performance of the MPF are presented. Since the algorithm is stochastic several runs of each flight were performed in order to receive a statistically valid result. Also featured is a comparison with the NINS algorithm and the ordinary MPF.

4.1 Filter Design

A lot of different parameters need to be chosen. The number of particles, process noise, measurement noise, variable map covariance and initial values for the particle filter will be treated in the following sections.

4.1.1 Number Of Particles

The number of particles determines how fast the algorithm will converge and how good the estimate will be. See Figure 4.1 here the performance of the algorithm when using different number of particles is plotted. Even at the lowest number of particles the filter converged every run, but performance improved when increasing the number. As shown in the figure no further improvements of the performance is obtained when using more than 5000 particles, which will be used in the sequel.

4.1.2 Process Noise

The added process noise, $Q_{t}^{add}$, controls the spread of the particles. The amount of information in the height database determines how well the algorithm manage to concentrate the particles. This balance is controlled by $Q_{t}^{add}$. A small $Q_{t}^{add}$
may cause the algorithm to lose the real state. On the other hand a large $Q^\text{add}_t$ will cause a larger covariance in the estimate. In the beginning of every flight, when
the algorithm has not yet converged, a larger $Q^\text{add}_t$ usually is applied but when the
algorithm has converged this impairs the effectiveness. A variable $Q^\text{add}_t$ can solve
this and a good approach is to use the estimated covariance of the particle filter,
$P_t^p$, as discussed in section 3.4.2. This gives us

$$Q^\text{add}_t = \kappa P_t^p,$$

where $\kappa$ is an empirical constant chosen to be $2 \cdot 10^{-3}$ for this application.

As a test the smallest possible added process noise when using 10000 particles
were decided. The criteria for ‘smallest possible’ was when all runs converged, and
this turned out to be when $\kappa = 10^{-4}$. The estimation error and covariance of the
estimation for this smaller $\kappa$ was then compared to the result when using the $\kappa$
chosen for this application. The comparison was performed to see if our chosen $\kappa$
introduced too much covariance in the estimate, but no significant difference was
found.

4.1.3 Measurement Noise

As discussed in section 2.3.1 the measurement noise is a combination of two Gaussians. This error is modelled according to

$$p_{e_t} = \Pr(\lambda_1)\mathcal{N}(m_1, \sigma_1) + \Pr(\lambda_2)\mathcal{N}(m_2, \sigma_2)$$

$$= 0.75 \cdot \mathcal{N}(0, 3^2) + 0.25 \cdot \mathcal{N}(12, 6^2),$$

(4.2)
where $Pr(\lambda_1)$ and $Pr(\lambda_2)$ are the probabilities of measuring the clearance to the ground and to the tree tops respectively. The numbers are from [12]. The measurement error pdf is depicted in Figure 4.2.

![Figure 4.2. Measurement error for the radar altimeter.](image)

The measurement noise above is based on collected data from several flights. In an attempt to optimize the measurement error, data from the first flight used in this thesis was collected. From this flight the true measurement noise was extracted and the MPF was run on this flight with both the extracted measurement noise and the assumed measurement noise. A comparison showed that the performance of the MPF was the same for both noise models.

### 4.1.4 Map Uncertainty

The measurement noise from section 4.1.3 also depends on the errors introduced by the height database. We learned from section 2.3.2 that this can be modelled by adding additional covariance to the measurement noise. Frykman [5] discusses the use of adding both a constant and a variable map covariance to the measurement noise, denoted by $R_{\text{const}}$ and $R_{\text{var}}$ respectively. The idea behind adding both of them in contrast to adding only constant map covariance is the fact that when interpolating in a very rough, highly non-linear, terrain the uncertainty of the interpolation is larger than if the terrain would have been flat. This larger uncertainty is represented by $R_{\text{var}}$. The added variable covariance can simply be set to the variance of the height database in a grid of size 500 m $\times$ 500 m around
the estimated position. The measurement noise from section 4.1.3 now becomes

\[
p_e = \Pr(\lambda_1)N(m_1, \sigma_1 + R_{\text{const}} + R_{\text{var}}) + \Pr(\lambda_2)N(m_2, \sigma_2 + R_{\text{const}} + R_{\text{var}}),
\]

(4.3)

where \(R_{\text{const}} = 8^2\) in this thesis.

4.1.5 Initial Values

\(x^g\) and \(x^k\) are both estimated with Kalman filters so their initial values will just be set to zero. The initial covariance for \(x^g\) and \(x^k\) are taken from Nordlund and Gustafsson [12] and are presented in Table 4.1. Furthermore, regarding the particles in the particle filter, the simplest way is to sample the initial particles uniformly in a rectangle. When the aircraft is on the runway the initial position is known with good accuracy, which means that the rectangle does not have to be that large. The values are from [12] and are chosen in a way that the standard deviation of the rectangle is 1 km in both directions.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Numerical value</th>
</tr>
</thead>
<tbody>
<tr>
<td>(N)</td>
<td>5000</td>
</tr>
<tr>
<td>(p(x^g_0))</td>
<td>(U\left(-\frac{1000\sqrt{3}}{r_0}, \frac{1000\sqrt{3}}{r_0}\right)) (U\left(-\frac{1000\sqrt{3}}{r_0 \cos L_0}, \frac{1000\sqrt{3}}{r_0 \cos L_0}\right))</td>
</tr>
<tr>
<td>(p(x^k_0))</td>
<td>(N(0, P_0))</td>
</tr>
<tr>
<td>(P_0)</td>
<td>(\text{diag}(50, 1, 1, 0.05\pi, 0.05\pi, 0.1\pi, 0.1\pi, 5 \cdot 10^{-3}, 5 \cdot 10^{-3})^2)</td>
</tr>
<tr>
<td>(p(u_t))</td>
<td>(N'(0, Q_t))</td>
</tr>
<tr>
<td>(Q_t)</td>
<td>(\text{diag}(0.2, 10^{-4}, 10^{-4}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6}, 10^{-6})^2)</td>
</tr>
<tr>
<td>(R)</td>
<td>(8^2)</td>
</tr>
<tr>
<td>(N_{\text{th}})</td>
<td>(\frac{2N}{\pi})</td>
</tr>
</tbody>
</table>

4.1.6 Evaluation Methods

As mentioned earlier the applied algorithm has a stochastic nature. To get a statistically valid result each flight was run 100 times. From these runs the root mean square error, \(RMSE_t\), and the mean of the covariance, \(P_{t\text{mean}}\) were computed according to

\[
RMSE_t = \left(\frac{1}{M} \sum_{i=1}^{M} \left\| \hat{x}^{(i)}_t - x^{\text{true}}_t \right\|_2 \right)^{1/2},
\]

(4.4a)

\[
P_{t\text{mean}} = \frac{1}{M} \sum_{i=1}^{M} P^{(i)}_t,
\]

(4.4b)
where $M$ is the number of runs. $P_t^{\text{mean}}$ is compared to the Cramér-Rao Lower Bound (CRLB). By making this comparison an indication of the algorithms performance is obtained. The true states are obtained by differential GPS (DGPS). The DGPS uses GPS signals together with a correction signal from receivers with known positions.

### 4.2 Flight Trajectories

To illustrate the performance of the MPF the result from two different flights are presented in this thesis. The first flight is depicted in Figure 4.3. This flight is rather simple for the algorithm. The flight only has two troublesome parts. The first difficulty is the occasion where the aircraft flies in circles. This is difficult because of the aircraft’s roll angle, which is too large for the radar altimeter to give measurements of the ground clearance. The second difficulty is when the aircraft is above the sea, where the height database can not provide any useful information for the MPF.

![Flight path](image)

**Figure 4.3.** The flight path of the first flight. The flight starts at the lake Roxen in the upper-left corner. To the right is the Baltic Sea.

The second flight will tell how well the MPF behaves when data is lacking for longer periods. The flight is depicted in Figure 4.4. During this flight the radar altimeter is manually shut off four times. Each time the radar altimeter is off for about eight minutes.
4.3 Results

The remaining sections of this chapter present the results from the test flights. Initially there is a comparison between an ordinary MPF and the extended MPF. Secondly, the performance of the extended MPF is analysed further. Finally, the extended MPF will be compared to the NINS algorithm.

4.3.1 Ordinary MPF - Extended MPF

As mentioned in section 3.6 a better estimation of the altitude should lead to a better estimation of the horizontal position. We will now compare the extended MPF described in section 3.6 with the ordinary MPF where the altitude state is in the particle filter instead of the GPBI filter.

By adding one more dimension in the non-linear state vector more particles will be needed. A similar test as the one in section 4.1.1 shows that adding more than 20000 particles does not improve performance significantly. This is shown in Figure 4.5.

The ordinary MPF was tested on the first flight and the result from the altitude estimation have been plotted in Figure 4.6 together with the results from the extended version. The extended MPF has a lower RMSE when data is missing, otherwise the performance of both filters are similar. The ordinary MPF though has a slightly better covariance in the altitude channel. The GPBI filter suffer from loss of information due to the independence of mode history and this may explain why the ordinary MPF has a better altitude covariance.
4.3 Results

Figure 4.5. When estimating the altitude state in the particle filter more than 5000 particles are needed. This thesis will use 20000 particles for this purpose.

Figure 4.6. The top plot shows the altitude error and the bottom shows the altitude covariance for the extended MPF (solid) and the ordinary MPF (dashed).

Since we know that the $RMSE_t$ in the altitude estimation was similar for the two filter, it is expected that the horizontal performance should be similar as well.
This is exactly what Figure 4.7 shows. The only difference between the two filters is that the $RMSE_t$ of the extended MPF is smaller when data is missing. This should not come as a surprise, since the extended MPF during this time period had a smaller $RMSE_t$ of altitude. As stated before, a better estimation of the altitude should lead to a better estimation of the horizontal position.

Figure 4.7. The top plot shows the horizontal error and the bottom shows the horizontal covariance for the extended MPF (solid) and the ordinary MPF (dashed).

Now, what about the computational costs? Going from 5000 particles to 20000 means an increase by a factor of four. The computational cost can be considered proportional to the number of particles, which means that the extended MPF is about four times faster than the ordinary MPF. Combined with the fact that no significant enhancement of performance shows when using an ordinary MPF, the extended MPF seems to be the better alternative. In the sequel only the extended MPF will be considered and will just be denoted MPF.

4.3.2 MPF

This section will show the result when applying the MPF on the two flights. Figure 4.8 shows the result from the first flight. The figure shows that both the $RMSE_t$ and $P_{t}^{mean}$ increases between 400 and 800 seconds and between 1000 and 1200 seconds. During these two periods the algorithm receives no new data due to the high roll angle and the flight above the sea, respectively. The algorithm stabilizes after approximately 50 seconds, which can be seen both after takeoff and after the two periods when data is missing. The $P_{t}^{mean}$ in the horizontal position is not as good as the CRILB. This could be the effect of the added process noise, $Q_t^{dd}$, although the result from section 4.1.2 does not indicate this. The $RMSE_t$
of the altitude estimate is stable throughout the whole flight but we can see how the $P^\text{mean}_t$ increases when data is missing. We also see that the $P^\text{mean}_t$ in the altitude channel differs more from the CRLB than in the horizontal estimation. From section 4.3.1 we saw that the $P^\text{mean}_t$ could be improved with the ordinary MPF, but this would not improve the $\text{RMSE}_t$.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure48.png}
\caption{The result from the first flight. The $\text{RMSE}_t$ (solid), the $P^\text{mean}_t$ (dashed) and the CRLB (dash-dotted). The plot on the top shows the result in the horizontal direction and the bottom plot show the result for the altitude channel.}
\end{figure}

The second flight puts the algorithm more to the test. During this flight the pilot shuts the radar altimeter off. This is done four times for about eight minutes each time. This flight will evaluate how well the algorithm recovers after longer periods of missing data. The result is shown in Figure 4.9. The algorithm recovers in just a few seconds and stabilizes in approximately 50 seconds after every period of missing data. One exception occurs after the last time the radar altimeter was shut down, where the $\text{RMSE}_t$ in the horizontal position does not quite achieve the same performance as before. Otherwise the result is still basically as for the first flight, in horizontal as well as in vertical position.

4.3.3 NINS

NINS is usually run with INS, TAP and GPS. Not many flights are performed without the support of GPS. The second flight however is performed solely with INS and TAP, which makes a fair comparison between the MPF and NINS possible. Unfortunately, the altitude estimation from NINS was not recorded which means that only a comparison in horizontal position will be made in this thesis.
Figure 4.9. The result from the second flight. The $RMSE_t$ (solid), the $P_t^{mean}$ (dashed) and the CRLB (dash-dotted). The plot on the top shows the result in the horizontal direction and the bottom plot show the result for the altitude channel.

Figure 4.10 shows the $RMSE_t$ for the extended MPF and the estimation error of NINS. The figure shows that the overall performance of NINS is better than the performance of the MPF. The estimation error is smaller when using NINS. Worth noticing is that NINS performs better than the MPF when data is missing. Why this is will not be pursued further in this thesis but we need to keep in mind that NINS has been under development for several years and can be assumed to be tuned in to perform at its best. In one aspect, though, the MPF is better than NINS; the MPF needs less time to recover from data losses.

4.4 Summary Of Chapter 4

In this chapter the performance of the extended MPF from section 3.6 was evaluated. Comparisons were made with an ordinary MPF with respect to computational costs and performance gain. The comparison showed that the extended MPF has the same performance but for less computational cost. The extended MPF was also compared with the NINS algorithm, where NINS showed better performance. On the other hand, the extended MPF recovered faster from periods of data loss.
Figure 4.10. The estimation error of the MPF (solid) and NINS (dashed).
Chapter 5

Conclusions And Future Work

In this thesis a sensor fusion algorithm based on Monte Carlo methods have been tested on real flight data. The results from Chapter 4 give us an indication of how well this algorithm perform. This chapter will discuss the potential of the algorithm and give some suggestions on future work.

5.1 Conclusions

Marginalization, or Rao-Blackwellization, allows the algorithm presented in this thesis to combine the particle filter with the kalman filter and the GPB1 filter. The three filters are applied where they perform the best and this combination yields a good mixture between a computationally tractable and a close to optimal solution.

This algorithm was compared with an ordinary mpf. This comparison showed that no significant losses are introduced when estimating the altitude state in the GPB1 filter instead of the particle filter.

The comparison between this algorithm and the NINS algorithm showed similar performance, although the NINS algorithm was slightly better. The MPF recovered faster from periods of data less than the NINS algorithm, but NINS had much better performance when data was missing. Here we need to keep in mind that NINS has been under development for a long time whereas the algorithm presented in this thesis still is in its infancy.

In order to make the algorithm work as intended, it needs time to converge. During this time good terrain information is a necessity.

5.2 Future Work

The most time consuming event in this algorithm is database queries. For each particle one database lookup is performed, each iteration. Thanks to marginalization the number of particles, and therefore the computational load induced by
the MPF, decreases drastically. Before system implementation becomes a topical question, more studies needs to be done concerning the work load on a system computer. In a field like aircraft navigation, reliable estimates is of great importance. The robustness of the algorithm is one thing that needs to be analyzed further, but also how to handle lack of data. This thesis has merely brushed against the topic of a malfunctioning radar altimeter. How to handle a malfunctioning sensor, i.e. fault detection, should be pursued further.
Bibliography


Appendix A

Continuous State Space Model

$A(t)$ is given by

$$A(t) = \begin{bmatrix}
A_n^0(t) & 0_{2 \times 1} & A_l^0(t) \\
0_{1 \times 2} & 0 & 0_{1 \times 7} \\
A_n^l(t) & 0_{7 \times 1} & A_l^l(t)
\end{bmatrix}$$

where

$$A_n^0(t) = \begin{bmatrix}
0 & 0 \\
\frac{1}{r_0 \cos L} \sin L & 0
\end{bmatrix},$$

$$A_l^0(t) = \begin{bmatrix}
\frac{1}{r_0} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & \frac{1}{r_0 \cos L} & 0 & 0 & 0 & 0 & 0
\end{bmatrix},$$

$$A_n^l(t) = \begin{bmatrix}
d^y_n & d^v_n & d^x_n & 0 & d^\gamma_n & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}^T,$$

$$A_l^l(t) = \begin{bmatrix}
de^y_n & de^v_n & -f_z & 0 & f_m & 0 & e_{b,11} & e_{b,12} \\
0 & 0 & -\frac{1}{r_0} & d^y_n & d^c_n & 0 & 0 \\
\frac{1}{r_0} & 0 & d^y_n & 0 & d^\gamma_n & 0 & 0 \\
0 & \tan \beta & d^\gamma_n & d^2 \gamma_n & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}^T,$$
and

\[
\begin{align*}
\dot{v}_n^a &= -\frac{v_e^2}{r_0 \cos^2 L} + 2\omega_{ie} v_n \cos L \\
\dot{v}_n^e &= \frac{v_n v_e}{r_0 \cos^2 L} - 2\omega_{ie} v_d \sin L + 2\omega_{ie} v_n \cos L \\
\dot{v}_n^\gamma &= \omega_{ie} \sin L \\
\dot{v}_e^a &= \frac{v_e}{r_0} \\
\dot{v}_e^\gamma &= \frac{v_e \tan L}{r_0} + \omega_{ie} \cos L \\
\dot{v}_n^a &= -2\frac{v_n \tan L}{r_0} - 2\omega_{ie} \sin L \\
\dot{v}_n^e &= \frac{v_n \tan L}{r_0} + 2\omega_{ie} \sin L \\
\dot{v}_n^\gamma &= \frac{v_n \tan L}{r_0} + \frac{v_d}{r_0} \\
\dot{v}_e^a &= \frac{v_e}{r_0} \\
\dot{v}_e^\gamma &= \frac{v_e \tan L}{r_0}, \quad \dot{v}_n^\gamma = -\dot{v}_n^e \\
\dot{v}_e^\gamma &= \frac{v_e}{r_0}, \quad \dot{v}_n^\gamma = -\dot{v}_n^e \\
\dot{v}_e^\gamma &= \omega_{ie} \cos L + \frac{v_e \tan L}{r_0}, \quad \dot{v}_n^\gamma = -\dot{v}_n^e.
\end{align*}
\]

\(B(t)\) is given by

\[
B(t) = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & c_{b,11}^n & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & c_{b,12}^n & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & c_{b,21}^n & c_{b,22}^n & 0 & 0 & 0 & 0 & 0 \\
0 & c_{b,11}^n & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & c_{b,12}^n & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & c_{b,21}^n & c_{b,22}^n & c_{b,23}^n & 0 & 0 & 0 & 0 \\
0 & c_{b,31}^n & c_{b,32}^n & c_{b,33}^n & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]
Upphovsrätt

Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under 25 år från publiceringsdatum under förutsättning att inga extraordinära 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 icke-kommersiell 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 nämd 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/](http://www.ep.liu.se/)

Copyright

The publishers will keep this document online on the Internet — or its possible replacement — for a period of 25 years 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 his/her 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/](http://www.ep.liu.se/)

© Tomas Hektor