Real Time Implementation of Map Aided Positioning using a Bayesian Approach

Examensarbete utfört i Reglerteknik
vid Tekniska Högskolan i Linköping
av

Niklas Svenzén

Reg nr: LiTH-ISY-EX-3297-2002
Real Time Implementation of Map Aided Positioning using a Bayesian Approach

Examensarbete utfört i Reglerteknik vid Tekniska Högskolan i Linköping av

Niklas Svenzén

Reg nr: LiTH-ISY-EX-3297-2002

Supervisor: Peter Hall, NIRA Dynamics
Rickard Karlsson, LiTH

Examiner: Fredrik Gustafsson

Linköping, 2nd December 2002.
With the simple means of a digitized map and the wheel speed signals, it is possible to position a vehicle with an accuracy comparable to GPS. The positioning problem is a non-linear filtering problem and a particle filter has been applied to solve it. Two new approaches studied are the Auxiliary Particle Filter (APF), that aims at lowering the variance of the error, and Rao-Blackwellization that exploits the linearity in the model. The results show that these methods require problems of higher complexity to fully utilize their advantages.

Another aspect in this thesis has been to handle off-road driving scenarios, using dead reckoning. An off-road detection mechanism has been developed and the results show that off-road driving can be detected accurately. The algorithm has been successfully implemented on a hand-held computer by quantizing the particle filter while keeping good filter performance.
Abstract

With the simple means of a digitized map and the wheel speed signals, it is possible to position a vehicle with an accuracy comparable to GPS. The positioning problem is a non-linear filtering problem and a particle filter has been applied to solve it. Two new approaches studied are the Auxiliary Particle Filter (APF), that aims at lowering the variance of the error, and Rao-Blackwellization that exploits the linearities in the model. The results show that these methods require problems of higher complexity to fully utilize their advantages.

Another aspect in this thesis has been to handle off-road driving scenarios, using dead reckoning. An off road detection mechanism has been developed and the results show that off-road driving can be detected accurately. The algorithm has been successfully implemented on a hand-held computer by quantizing the particle filter while keeping good filter performance.

Keywords: Rao-Blackwellization, Bayesian Estimation, Map Matching, Particle Filter, Monte Carlo Simulation, Positioning, Off-road Driving, Quantization, Fixed Point Arithmetics, Modeling.
Acknowledgment

This master’s thesis brings an end to my Master of Science Degree in Applied Physics and Electrical Engineering at Linköpings Universitet, Institute of Technology. During my thesis I have had help and input from several people, and they deserve to be acknowledged.

I would like to express my gratitude towards my supervisors, M.Sc. Peter Hall and lic. Rickard Karlsson for their help and guidance during this thesis. I would also like to thank my examiner Professor Fredrik Gustafsson for valuable discussions and inspiring ideas. Many thanks to Stefan Ahlqvist for helping me out with the NVP and Greger Cronquist who helped me with the fixed point arithmetics. Finally I would like to thank the staff at NIRA Dynamics for making me feel welcome during my work there.

Linköping, December 2002
Niklas Svenzén
Notation

In this section the abbreviations and symbols that are used in this thesis are explained.

Symbols

- $\delta_{rl}$: The difference in radius between the rear right and the rear left tire.
- $\delta_{rn}$: Nominal radius offset.
- $K_{mu,t}, K_{tu,t}$: Kalman gain matrix for the measurement update and the time update respectively.
- $N$: Number of samples used for parameter estimation.
- $N(\mu, \sigma)$: Normal distribution with mean $\mu$ and standard deviation $\sigma$.
- $p(x)$: Probability density function (pdf) for the stochastic variable $x$.
- $p(x, y)$: Joint density for the stochastic variables $x$ and $y$.
- $p(x|y)$: Conditional density for $x$ given a realization of $y$.
- $\psi$: The vehicle's angle of direction.
- $Q^p_t, Q^k_t$: Process noise covariance matrix for the particle and Kalman states respectively.
- $R_t$: Measurement noise covariance matrix.
- $\sigma$: Standard deviation.
- $T_s$: Sample time.
- $\mathbb{U}(a, b)$: Uniform distribution on the interval $[a, b]$.
- $w^{(i)}$: Probability weight, or importance weight, for particle $i$.
- $x$: Denotes the entire state space or the longitudinal position of the vehicle, which one should be clear from the context.
- $\hat{x}$: Estimate of the stochastic variable $x$.
- $x^p_t$: The particle filter part of the state space.
- $x^k_t$: The Kalman filter part of the state space.
- $y$: The latitudinal position of the vehicle or measurement if no risk for confusion.
- $y_{meas}$: Measurement if confusion is possible.
- $\mathbb{Y}_t$: Measurements made up to and including time $t$.
- $z(t)$: Time continuous vector.
- $z_t$: Time discrete vector.
Operators and functions

\[ x^{(i)}_t \sim N(0, \sigma) \quad x^{(i)}_t \text{ is a realization of, or a sample drawn from, the distribution } N(0, \sigma) \]

\[ E\{x\} \quad \text{Expectancy of a stochastic variable } x. \]

\[ \|x - y\|_2 \quad \text{The Euclidean distance between the vectors } x \text{ and } y. \]

\[ F_t(x), G_t(x) \quad \text{State transition functions.} \]

Abbreviations

3G The 3rd Generation of cellular telephones.

ABS Anti-Lock Braking System.

APF Auxiliary Particle Filter.

CAN Controller Area Network

CEP Circular Error Probable

CoG Center of Gravity

GPS Global Positioning System.

GUI Graphical User Interface.

i.i.d. Independent Identically Distributed

MAP Map Aided Positioning.

pdf Probability Density Function

RB Rao-Blackwellization

SIS Sequential Importance Sampling

SIR Sampling Importance Resampling, also known as Bayesian bootstrap
Contents

1 Introduction .................................. 1
  1.1 Background .................................. 1
  1.2 Principles of Positioning ................. 2
    1.2.1 Relative Positioning .................. 2
    1.2.2 Absolute Positioning ................. 2
    1.2.3 Implementation ....................... 2
  1.3 NIRA Dynamics ............................ 3
  1.4 Problem Specification ...................... 4
  1.5 Objectives ................................ 4
  1.6 Thesis Outline ............................. 4

2 Bayesian Estimation .......................... 5
  2.1 Estimation Concepts ....................... 5
  2.2 Recursive Bayesian Estimation .......... 6
    2.2.1 Implementation ....................... 7
    2.2.2 Particle Filter ....................... 8
    2.2.3 The Auxiliary Particle Filter ...... 13

3 Rao-Blackwellization ........................ 15
  3.1 Partitioning of the State Space Model .... 15
    3.1.1 The Gain in Using Rao-Blackwellization 17

4 Map Aided Positioning ........................ 19
  4.1 Obtaining Information From the Map ...... 21
    4.1.1 Finding the Closest Road Segment .... 21
    4.1.2 Including Traffic Information ...... 24
  4.2 The State Space Model .................... 24
    4.2.1 Discretization ........................ 26
    4.2.2 Noise Models ......................... 28
    4.2.3 Observability ......................... 29
    4.2.4 Evaluation of the State Space Model 29
  4.3 Evaluating the Various Filters .......... 34
    4.3.1 The Auxiliary Particle Filter ....... 34
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.2 The Rao-Blackwellized Filter</td>
<td>36</td>
</tr>
<tr>
<td>4.3.3 Tuning</td>
<td>38</td>
</tr>
<tr>
<td>5 Filter Robustness and Off-road Driving</td>
<td>41</td>
</tr>
<tr>
<td>5.1 Off-road Detection</td>
<td>41</td>
</tr>
<tr>
<td>5.2 Dead Reckoning</td>
<td>42</td>
</tr>
<tr>
<td>5.2.1 Several Estimates of the Dead Reckoning Position</td>
<td>43</td>
</tr>
<tr>
<td>5.2.2 Summary on Detection</td>
<td>44</td>
</tr>
<tr>
<td>5.3 Finding the Road Again</td>
<td>44</td>
</tr>
<tr>
<td>5.3.1 Off-road When Using GPS Support</td>
<td>46</td>
</tr>
<tr>
<td>5.3.2 Results</td>
<td>46</td>
</tr>
<tr>
<td>6 Optimization</td>
<td>51</td>
</tr>
<tr>
<td>6.1 Quantization</td>
<td>51</td>
</tr>
<tr>
<td>6.2 Fixed Point Representation</td>
<td>52</td>
</tr>
<tr>
<td>6.3 Bottlenecks in the Algorithm</td>
<td>54</td>
</tr>
<tr>
<td>6.4 Implementation</td>
<td>55</td>
</tr>
<tr>
<td>6.4.1 Results from the Quantized Particle Filter</td>
<td>55</td>
</tr>
<tr>
<td>7 Conclusions and Future work</td>
<td>57</td>
</tr>
<tr>
<td>7.1 Conclusions</td>
<td>57</td>
</tr>
<tr>
<td>7.2 Future Work</td>
<td>58</td>
</tr>
<tr>
<td>Bibliography</td>
<td>60</td>
</tr>
<tr>
<td>A Spring Tire Model</td>
<td>61</td>
</tr>
<tr>
<td>B Acceleration Noise</td>
<td>63</td>
</tr>
<tr>
<td>C Testruns</td>
<td>65</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Background

Telematics and vehicle positioning are today a growing market. It is more and more common that new cars come with a complete navigation system. Close to all of these systems are based on the Global Positioning System (GPS), that is dependent on reliable satellite communications. This can be a problem for instance in dense urban areas where buildings cause multipath distortion or in bad weather conditions when the satellite signals can be disrupted. Another issue is that GPS receivers are high-technology utilities that still are quite expensive. In this master’s thesis a different approach to navigation is taken.

By measuring the wheel speeds of the car, the relative speed and yaw rate can be estimated. By mapping the movements onto a map, using map-matching, the absolute position can be obtained. The wheel speed signals are taken from the ABS-sensors (Anti-Lock Braking System), that is available in all relatively new cars. In modern cars, these measurements are available on the internal Controller Area Network (CAN), so there is no need for expensive extra equipment, thus making the positioning solution cheap. The only external reference needed is an accurate digital map of the road network.
1.2 Principles of Positioning

This section is an introduction to the basic principles of positioning, where the concepts of relative and absolute positioning are defined.

1.2.1 Relative Positioning

To be able to calculate the position of a vehicle some form of outer observations or measurements have to be made. For instance, if the velocity of the vehicle can be measured, the position in the next instance can be calculated by just adding the relative movement to the current position. This is a relative positioning scheme, since the calculations have to be initialized with an absolute position. Due to errors in the velocity measurement, the position estimate will suffer from drifts, that is errors that will increase with time.

1.2.2 Absolute Positioning

Starting with a relative position, some extra information has to be provided to get an absolute position. Here, this extra information is a geographical map containing precise street information. The basic idea behind absolute positioning using map information is best explained by an example. Consider a vehicle that is driving straight ahead. The vehicle is then bound to be on a road that goes straight ahead so all the turns on the map are impossible candidates for the true position. Now assume that the vehicle turns left and then quickly to the left again. The number of roads that first heads straight ahead, then turns left and thereafter turns left again is further limited. Finally, continuing in the same way, there will only be one possible trajectory that fits the road network of the map and the vehicle can then be positioned onto this road (see Figure 1.1). To avoid having to search through the entire world map some form of initial position still has to be provided, but the uncertainty in this initial position can be relatively large (∼ 1 km²). Another feature of this approach is that the drifts are accounted for. Every time the vehicle turns, the true position is most likely to be in an intersection, so a form of self-calibration can be made.

1.2.3 Implementation

The above description is simplified for clarity, but the main principles are still valid. The implementation of the absolute positioning method described has to take into account that the map does not always fully agree with the real world. The vehicles is not always on the road or the road might not be on the map. Another problem to take into consideration is that the measurements of the speed and angular velocity of the vehicle are a bit noisy.
The figures (a)-(f) show the evaluation of the possible vehicle positions. (a) shows the initial distribution where all roads are equally likely, (b) is the possible positions after the first left turn. In (e) and (f) the filter has almost converged and should the vehicle turn right after (f), there would only be one possible position.

Since the position of the vehicle is estimated by following different roads and then filtering out the least likely ones, a form of recursive method such as a sequential Monte Carlo filter seems well suited for this problem. A master’s thesis in this area has already been conducted, see [13], and this thesis is meant to further develop and fine tune the solutions presented there.

1.3 NIRA Dynamics

This master’s thesis has been carried out at NIRA Dynamics in Linköping. NIRA Dynamics is a research and development company focusing on vehicle dynamics and active safety products based on state-of-the-art sensor fusion and control technology. For more information about the company, visit their web site, http://www.niradynamics.se.
1.4 Problem Specification

Positioning is determining the exact location, or coordinates, of for instance a vehicle. In order to do this, the position has to be given relative a reference frame, like a map. The simple idea behind the Map Aided Positioning (MAP) algorithm is to compare trajectories given by dead reckoning with the road network in a digitized map, by choosing the closest match the absolute position can be found. This makes up a highly non-linear filtering problem and therefore a particle filter has been designed to solve it. A particle filter is computationally demanding and since the MAP algorithm should run in a real time environment the existing particle filter needs to be improved.

The improvements that need to be made are a robustifications of the existing algorithm, since the algorithm sometimes diverges, an optimization to speed up the algorithm to be able to run it as an online application, and finally an off-road handling mechanism that will handle the scenario where the vehicle drives off the road network of the map.

1.5 Objectives

The problem consists of two parts:

- **Robustification.** The existing MAP algorithm sometimes diverges and gives erroneous results. This is mainly due to map incorrectnesses and off-road drive cases. Therefore the algorithm needs to be more robust against map imperfections and it should be investigated how to handle off-road driving, that is when the vehicle drives off the road network.

- **Optimization.** Decreasing the calculation burden for the MAP algorithm is essential to get it running in real time. One aspect is to make the algorithm more robust so that less resources are needed to solve the problem. Another aspect is to implement the algorithm using fixed point arithmetics to speed up the entire process.

1.6 Thesis Outline

The report starts out describing the basics of Bayesian estimation that the whole algorithm is based on. Also some different particle filters are reviewed and the auxiliary particle filter is introduced. Chapter 3 explains the concept of Rao Blackwellization. In Chapter 4, the theory is implemented into a map aided positioning algorithm. Theories on how to handle off-road driving without map information is presented in Chapter 5. Chapter 6 deals with quantization and fixed point arithmetics. Conclusions are drawn and future work is suggested in Chapter 7.
Chapter 2

Bayesian Estimation

This chapter describes the fundamentals of filtering and how the filtering problem could be solved using a Bayesian approach. As will be shown, the implementation of Bayesian estimation needs to be based on some form of numerical approximation and this will end up in the use of Sequential Monte Carlo methods, also known as particle filters. In the end of this chapter some approaches to decrease the variance of the error in order to decrease the computational burden will be presented.

2.1 Estimation Concepts

First of all, the concept of estimation has to be defined. Consider the problem of finding the value of the parameter $x(t)$ at a certain time $t$. The simplest method is of course to measure the parameter directly, but this is often not possible, and the measurement also has a degree of uncertainty or noise $e(t)$. Suppose that measurements $y(t) = h(x(t), e(t))$ are sampled with a sample time $T_s$. The measurements up to time $t$ will then be $\mathcal{Y}_t = \{y(kT_s), k = 0, ..., t/T_s\}$. The estimation problem is then to find an estimate $\hat{x}_t = \hat{x}_t(\mathcal{Y}_t)$.

There are two approaches to this problem, according to [21]. One is to consider the parameter $x$ to have a specific, but unknown, value. Another is to treat $x$ as a stochastic variable with a prior probability density function (pdf) $p_t(x)$. The latter way is also known as the Bayesian approach and this is the approach to be taken in this thesis. It is called Bayesian, due to the extensive use of Bayes’ rule [4, p. 36]

$$p(x|y) = \frac{p(y|x)p(x)}{\int_{\mathbb{R}^n} p(y|x)p(x) \, dx} = \frac{p(y|x)p(x)}{p(y)}, \quad (2.1)$$

where $x, y$ are stochastic variables with probability density functions $p(x)$, $p(y)$ and with joint density $p(x,y)$. If $x$ is the (unknown) state of some system and $y$ are measurements of this system, then it is very interesting to know how likely a certain state $x$ is given the measurement $y$, $p(x|y)$. This is often impossible to
calculate, but the opposite, how likely a certain measurement $y$ is given the state $x$, $p(y|x)$ is more often an easy entity to compute. Bayes’ rule is the glue that binds these expressions together.

### 2.2 Recursive Bayesian Estimation

Assume that we have a state space system on the form:

\begin{align}
    x_{t+1} &= f(x_t, u_t) + v_t \quad (2.2a) \\
    y_t &= h(x_t) + e_t \quad (2.2b)
\end{align}

where $v_t$ and $e_t$ are additive independent noise signals with distributions $p_{v_t}$ and $p_{e_t}$ respectively. The focus for this thesis is the application of vehicle positioning. For instance let $x_t$ denote the vector that contains the position and orientation calculated from the wheel speeds $u_t$ and let $Y_t = \{y_t\}_{i=1}^t$ be the measurements up to time $t$. All that is known about the state of the system at time $t$, given the measurements $Y_t$, is given by the pdf $p(x_t|Y_t)$. Using Bayes’ rule (2.1), this can be rewritten as

\[
p(x_t|Y_t) = \frac{p(y_t|x_t, Y_{t-1})p(x_t|Y_{t-1})}{p(y_t|Y_{t-1})}. \tag{2.3}
\]

The density $p(y_t|x_t, Y_{t-1})$ is a measure of how likely the observation $y_t$ is given a certain state $x_t$. The likelihood in this example is $p_{e_t}(y_t - h(x_t))$. The equation above has a recursive form, so assuming that $p(x_t|Y_{t-1})$ is known then this pdf can now be calculated by inserting the model (2.2) into (2.3)

\[
p(x_t|Y_t) = \alpha^{-1}_t p_{e_t}(y_t - h(x_t))p(x_t|Y_{t-1}) \tag{2.4a}
\]

\[
\alpha_t = \int_{\mathbb{R}^n} p_{e_t}(y_t - h(x_t))p(x_t|Y_{t-1}) \, dx_t. \tag{2.4b}
\]

Equation 2.4 is known as the measurement update of the pdf and $\alpha_t = p(y_t|Y_{t-1})$ in (2.4b) can be regarded as a normalizing factor. What is left to calculate in order to complete the recursion step is the time update, that is $p(x_{t+1}|Y_t)$. This can be rewritten as a marginalization with respect to $x_t$

\[
p(x_{t+1}|Y_t) = \int_{\mathbb{R}^n} p(x_{t+1}, x_t|Y_t) \, dx_t. \tag{2.5}
\]

Using one of the the rules for joint densities, $p(x, y) = p(x|y)p(y)$, on the integrand in (2.5) yields

\[
p(x_{t+1}, x_t|Y_t) = p(x_{t+1}|x_t, Y_t)p(x_t|Y_t). \tag{2.6}
\]

Assuming that the input noises are white, then the state model is a Markovian process according to [21]. From this it follows that $p(x_{t+1}|x_t, Y_t) = p(x_{t+1}|x_t)$. 

2.2 Recursive Bayesian Estimation

The assumed model gives \( p(x_{t+1} | x_t) = p_{v_t}(x_{t+1} - f(x_t, u_t)) \). Inserting this into (2.6) and (2.5) the time update equation becomes

\[
p(x_{t+1} | Y_t) = \int_{\mathbb{R}^n} p_{v_t}(x_{t+1} - f(x_t, u_t)) p(x_t | Y_t) \, dx_t. \tag{2.7}
\]

To sum up, the Bayesian solution to the problem (2.2) is given by

\[
p(x_t | Y_t) = \alpha_t^{-1} p_{e_t}(y_t - h(x_t)) p(x_t | Y_{t-1}) \tag{2.8a}
\]

\[
p(x_{t+1} | Y_t) = \int_{\mathbb{R}^n} p_{v_t}(x_{t+1} - f(x_t, u_t)) p(x_t | Y_t) \, dx_t, \tag{2.8b}
\]

where \( \alpha_t = \int_{\mathbb{R}^n} p_{e_t}(y_t - h(x_t)) p(x_t | Y_{t-1}) \, dx_t \) is a normalization factor. From (2.8) it is possible to recursively compute \( p(x_t | Y_t) \) given the measurements \( Y_t \). The recursion has to be initialized with \( p(x_0 | Y_{-1}) = p(x_0) \), where \( p(x_0) \) is some representation of prior knowledge, for instance a uniform distribution.

The solution in (2.8) contains the distribution of the states given specific measurements. From the posterior an estimate can be obtained using various methods, for instance using the minimum mean square method (MMS) \[14\]:

\[
\hat{x}_{t}^{MMS} \triangleq \int_{\mathbb{R}^n} x_t p(x_t | Y_t) \, dx_t. \tag{2.9}
\]

In Bayesian estimation, the parameter is regarded as a random variable, but the estimate provides a specific value. The problem is that this value does not give any information about how good, or how probable this value is. In order to get control over this, the error correlation matrix, or the spread of the mean

\[
P_t \triangleq E((x_t - \hat{x}_t)(x_t - \hat{x}_t)^T), \tag{2.10}
\]

should be investigated. This entity is calculated from the posterior pdf by

\[
P_t = \int_{\mathbb{R}^n} (x_t - \hat{x}_t)(x_t - \hat{x}_t)^T p(x_t | Y_t) \, dx. \tag{2.11}
\]

2.2.1 Implementation

The solution (2.8) seems simple enough, but in reality it is often hard or impossible to solve it analytically. Especially when the measurement equations are non-linear or the noise distribution is non-Gaussian. A special case is when the system is linear and the noise distribution is Gaussian. An optimal solution is then given by the Kalman filter \[12, pp. 289–320\]. When implementing a solution, in general, some form of numerical approximation need to be considered. In \[7, 13\] a number of approximative solutions are presented. Methods of special interest are the sequential Monte Carlo methods and those are the ones that this master’s thesis will focus on.
2.2.2 Particle Filter

The main idea in Bayesian estimation is to compute the posterior density \( p(x_t|Y_t) \), since this contains all the information worth knowing about the state \( x_t \) given the measurements \( Y_t \). If this pdf is known, an estimate is easily obtained, for instance by calculating the mean value \( \int xp(x|Y) \, dx \). The sequential Monte Carlo methods simulate this distribution with a finite number of samples, or “particles”. The more particles, the better the approximation gets. Because of the concept of particles these filters are often referred to as particle filters. These filters are becoming increasingly popular and are used in a variety of applications [11].

In the vehicle positioning case we have the following non-linear time discrete system

\[
\begin{align*}
x_{t+1} &= f(x_t) + v_t \quad (2.12a) \\
y_t &= h(x_t) + e_t, \quad (2.12b)
\end{align*}
\]

where \( x_t \) is the state of the system, for instance the position and angle, and where \( y_t \) are measurements. The basic idea behind the particle filter is to approximate \( p(x_t|Y_t) \) with a set of \( N \) random samples \( \{x_t^{(i)}\}_{i=1}^N \), where each particle \( x_t^{(i)} \) is assigned a weight. The weight of each particle should in some way reflect how probable it is that the properties of this particle are the correct ones. Those particles with the highest weights are propagated in time and natural selection is performed. It should be noted that the number of particles, \( N \), has to be chosen large enough to represent the probability density function.

Monte Carlo Integration

A very common operations in Bayesian estimation is to calculate some form of expectancy, such as the MMS estimate (2.9) or the error covariance matrix (2.11). There is a special technique used to calculate integrals on this form when the exact distribution is not explicitly known. In fact, a way of solving these integrals with arbitrarily high precision is presented in [16]. Consider an integral on the form

\[
I = \int g(x)p(x) \, dx. \quad (2.13)
\]

This can be viewed as a calculation of the mean of the function \( g \) given the underlying pdf \( p \). If it is possible to draw samples from \( p \), then it is easy to approximate the integral by:

\[
\hat{I} = \frac{1}{N} \sum_{i=1}^N g(x^{(i)}), \quad (2.14)
\]

where \( \{x^{(i)}\}_{i=1}^N \) are independent identically distributed (i.i.d.) samples from \( p \). The accuracy depends on how many samples, \( N \), that are drawn. This technique is known as Monte Carlo integration [1, 13, 16]. In many cases, and especially in the vehicle positioning case, it is often not possible to draw samples from \( p \). The
solution is to use a proposal distribution $q$ from which samples can be drawn. Then the integral becomes:

$$I = \int g(x) \frac{p(x)}{q(x)} q(x) \, dx,$$

which is approximated by

$$\hat{I} = \frac{1}{N} \sum_{i=1}^{N} \frac{p(x^{(i)})}{q(x^{(i)})} g(x^{(i)}).$$

This time the i.i.d. samples $\{x^{(i)}\}_{i=1}^{N}$ are drawn from $q$. The weights $w_t = \frac{p(x_t)}{q(x_t)}$ are called importance weights and can be seen as a measure of how skew the distribution $q(x_t)$ is compared to $p(x_t)$. The sampling technique is known as importance sampling [6].

### Approximating the Solution

The integrals of interest often contain the pdf $p(x_t|Y_t)$. For instance the position estimate is calculated as

$$\hat{x}_t = \int_{\mathbb{R}^n} x_t p(x_t|Y_t) \, dx_t,$$

so this can be seen as the underlying pdf. Unfortunately it is hard to draw samples from this density function so instead it is rewritten as

$$p(x_t|Y_t) = \frac{p(y_t|x_t)}{p(y_t|Y_t)} \frac{p(x_t|Y_{t-1})}{w_t} \propto \frac{p(y_t|x_t)}{p(y_t|Y_{t-1})}.$$  

It is now possible to draw samples from the prior distribution $p(x_t|Y_{t-1})$ and the importance weights can be calculated according to

$$w_t^{(i)} = \frac{p(y_t|x_t^{(i)})}{w_t^{(i)}} w_{t-1} = p_t(y_t - h(x_t^{(i)})) w_{t-1}^{(i)} \quad \left\{ \text{i} = 1, \ldots, N \right\}.$$ 

The integrals of interest, (2.9) and (2.11), are now approximated with

$$\hat{x}_t^{MMS} \approx \sum_{i=1}^{N} \frac{\bar{w}_t^{(i)}}{w_t^{(i)}} x_t^{(i)}$$

$$P_t \approx \sum_{i=1}^{N} \frac{\bar{w}_t^{(i)}}{w_t^{(i)}} (x_t^{(i)} - \hat{x}_t)(x_t^{(i)} - \hat{x}_t)^T,$$

where $\{x_t^{(i)}\}_{i=1}^{N}$ are i.i.d. samples from $p(x_t|Y_{t-1})$ and $\bar{w}_t^{(i)}$ are also calculated with these samples. Figure 2.1 shows how the particle weights are propagated during a
turn. The left figure shows how the initial weights approximate the particle density in an intersection and the right shows the density after the turn is completed. The higher the peaks are, the more likely it is that the vehicle is located at this position.

Figure 2.1. The update of importance weights. The left figure depicts the initial distribution of particles in a road crossing. The right figure shows how the particle weights are updated after a right turn in the intersection.

Resampling

Importance sampling provides a tool to approximate the pdf, but after a while the importance weights may become very skewed, e.g. only a few particles are likely, whereas the rest have weights close to zero. If this is the case some form of resampling has to be made to keep the samples i.i.d. [6]. To sample from this distribution the particles are resampled according to their importance weights. Those particles that have high weights are more likely to survive, whereas those with lower weights are less likely. This way, the resampled samples together form an approximation of the pdf that the weights describe. The accuracy of the approximation can be made arbitrarily good by increasing the number of particles.
The resampling can be performed in a deterministic or in a stochastic way. An example of the latter is multinomial resampling [17], which is an efficient and often used resampling method. The method is to create an ordered set of numbers, $u_i$, uniformly distributed

$$u_i \in \mathbb{U}(0, 1), \quad i = 1, \ldots, N,$$

and then create the set of the cumulated weights according to

$$w_j = \sum_{k=1}^{j} a_i^{(k)}. \tag{2.22}$$

For each $i$ find the smallest $w_j$ that is larger than $u_i$ and keep the corresponding sample $x_i^{(j)}$, see Figure 2.2.

![Figure 2.2. Multinominal resampling. The asterisks represent the number of samples drawn from sample $i$. The dotted lines are samples drawn from $\mathbb{U}(0, 1)$.](image)

The problem caused by the resampling step is that the resulting samples are dependent since the samples are now offsprings from (perhaps) a few number of ancestors. This can give the effect that the particles are clustered into groups instead of representing the smooth probability density as they should. To lessen the dependency, some artificial noise is added to the resampled particles. The resampling step is necessary in order to avoid having the importance weights becoming too degenerated, but the resampling step does not have to be carried out in every iteration.
SIR Algorithm

The resampling idea is known as Bayesian Bootstrap or Sampling Importance Resampling (SIR) and the algorithm is as follows [6, 16].

1. Set \( t = 0 \) and generate \( N \) samples \( \{x_0^{(i)}\}_{i=1}^N \) from the initial density \( p(x_0) \).
2. Compute the weights \( w_t^{(i)} = p(y_t|x_t^{(i)}) \) and normalize: \( \bar{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^N w_t^{(j)}} \)
3. Calculate the estimate \( \hat{x}_t = \sum_{i=1}^N \bar{w}_t^{(i)} x_t^{(i)} \)
4. Generate a new set \( \{x_t^{(i)}\}_{i=1}^N \) by resampling with replacement \( N \) times from \( \{x_t^{(j)}\}_{j=1}^N \), with probability \( \Pr\{x_t^{(i)} = x_t^{(j)}\} = \bar{w}_t^{(j)} \)
5. Predict the new particles with a different noise realization for each particle. \( x_{t+1}^{(i)} = f(x_t^{(i)}) + v_t, \ i = 1, \ldots, N \)
6. Set \( t = t + 1 \) and repeat from 2.

The SIR algorithm resamples the particles at every iteration. This could be quite computationally expensive and the dependency between the samples becomes unnecessarily high, so there is another algorithm that updates the weights recursively, and only resamples when “necessary”, that is when the weights are too degenerated which implies that a lot of particles are highly unlikely. This algorithm is known as the Sequential Importance Sampling (SIS) [6, 16].

SIS Algorithm

As mentioned above, the SIS algorithm only resamples when the weights get too degenerated. To find out when this is the case, the effective sample size, \( N_{\text{eff}} \), that reflects the number of particles that contribute to the support of the probability density function, is studied. An approximation to the effective sample size is given in [6, 2]

\[
\hat{N}_{\text{eff}} \approx \frac{1}{\sum_{i=1}^N (w_t^{(i)})^2}.
\] (2.23)
If calculated this way, then $1 \leq \hat{N}_{\text{eff}} \leq N$, where $\hat{N}_{\text{eff}}$ is maximized when all weights have the same weight, that means no skewing, and is minimized when only one weight is non-zero, i.e. the samples are concentrated to a single spot. So when the samples are degenerated $\hat{N}_{\text{eff}}$ decreases and resampling should be performed when $\hat{N}_{\text{eff}}$ is less than a certain threshold $N_{\text{thres}}$. A value for the threshold suggested in [2] is $N_{\text{thres}} = \frac{2N}{3}$.

1. Set $t = 0$ and generate $N$ samples $\{x_0^{(i)}\}_{i=1}^N$ from the initial density $p(x_0)$ and set the weights $w_t^{(i)} = \frac{1}{N}$.

2. Update the weights $w_t^{(i)} = p(y_t | x_t^{(i)})\bar{w}_{t-1}^{(i)}$ and normalize: $\bar{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^N w_t^{(j)}}$.

3. If $\hat{N}_{\text{eff}} < N_{\text{thres}}$ or at every $k$:th iteration, generate a new set $\{x_t^{(i)}\}_{i=1}^N$ by resampling with replacement $N$ times from $\{x_t^{(j)}\}_{j=1}^N$, with probability $\Pr\{x_t^{(i)} = x_t^{(j)}\} = \bar{w}_t^{(j)}$. Reset the weights to $w_t^{(i)} = \frac{1}{N}$, $i = 1, \ldots, N$.

4. Predict the new particles with a different noise realization for each particle. $x_{t+1}^{(i)} = f(x_t^{(i)}) + v_t$, $i = 1, \ldots, N$.

5. Set $t = t + 1$ and repeat from 2.

2.2.3 The Auxiliary Particle Filter

In [13] particle filters were mainly implemented using the sampling importance re-sampling (SIR) and sequential importance sampling (SIS) methods. An alternative method proposed in [19, 16] is the Auxiliary Particle Filter (APF). This filter waits for an extra measurement and resamples those particles that are likely even after the new measurement. This yields a higher influence of particles with a large predictive likelihood and could be a good idea if the measurements contain a lot of information. It also means that the cost of sampling many times from particles with very low predictive likelihood is reduced since they will not survive to the second resampling. This improves the statistical efficiency of the sampling procedure and the total number of particles can be reduced according to [19]. Since only the most likely particles will survive, the artificial noise can be increased in order to give a more robust filter or the number of particles might be decreased keeping the same robustness. These properties seem very interesting for a real time application such as MAP. For more information on the Auxiliary Particle Filter, see [7, pp. 273–293].
APF Algorithm

1. Set $t = 0$ and generate $N$ samples $\{x_0^{(i)}\}_{i=1}^N$ from $p(x_0)$.

2. Set $w_0^{(i)} = \frac{1}{N}$, $k^{(i)} = i$, $i = 1, \ldots, N$

3. Predict the particles without noise. $\mu_t^{(i)} = \mu_{t-1}^{(i)}$

4. Compute the weights $\rho_{t-1}^{(i)} = p(y_t|\mu_{t-1}^{(i)}) \bar{w}_t^{(i)}$ and normalize: $\bar{\rho}_t^{(k)} = \frac{\rho_t^{(k)}}{\sum_{j=1}^N \rho_t^{(j)}}$

5. Calculate the effective sample size $\hat{N}_{\text{eff}}$

   $$\hat{N}_{\text{eff}} \approx \frac{1}{\sum_{i=1}^N (\bar{\rho}_t^{(i)})^2}$$

6. If $\hat{N}_{\text{eff}} < N_{\text{thres}}$ generate new indices $k^{(j)}$ by resampling from $\bar{\rho}_t^{(k)}$ for instance by using multinominal resampling.

   If no resampling, set $\bar{\rho}_t^{(i)} = \frac{1}{N}$

7. Predict the particles with different noise realizations.

   $x_t^{(j)} = f(x_{t-1}^{(j)}) + v_t$, $j = 1, \ldots, N$ and $v_t$ is Gaussian noise.

8. Update the weights $w_t^{(i)} = \frac{p(y_t|x_t^{(i)}) \bar{w}_t^{(i)}}{\bar{\rho}_{t-1}^{(i)}}$ and normalize: $\bar{w}_t^{(i)} = \frac{w_t^{(i)}}{\sum_{j=1}^N w_t^{(j)}}$

9. Set $t = t + 1$ and repeat from 3.

Since the APF algorithm uses an extra measurement update, some additional calculations compared to SIR and SIS have to be made. This algorithm is therefore somewhat more computationally expensive, but the decrease in variance as described in [19] should more than well compensate for this.
Chapter 3

Rao-Blackwellization

The particle filter solves the non-linear filtering problem in a numerical way. It is also true that the potential of the particle filter is higher the more non-linear the problem is. Obviously the complexity of the problem increases with the number of states to estimate. Therefore, with many states the number of particles needs to be very high in order to assure that all modes of the system are stimulated so that the filter performs well. A solution to this is proposed in [17, 18] and similar approaches are taken in [5, 6]. The essential idea is to estimate as much as possible of the problem analytically with a Kalman filter. The rest of the problem should now have a lower complexity and can be solved with a particle filter. This technique is known as Rao-Blackwellization. The purpose of Rao-Blackwellization is to reduce the variance of the estimate error, but it can also be used to get the same variance as before but using fewer particles to reduce the calculation burden.

3.1 Partitioning of the State Space Model

Let us start with general state space model such as

\[ x_{t+1} = f(x_t) + G_t(x_t)v_t \]
\[ y_t = h(x_t) + e_t. \]

Assume that the model (3.1) can be separated into a highly non-linear part, \( p \) (particle), and a more analytically tractable part, \( k \) (Kalman), so that \( x_t = \begin{pmatrix} x^p_t \\ x^k_t \end{pmatrix} \). Since the model was assumed to be non-linear in \( p \), the following model is a good base to work with

\[ x^p_{t+1} = f^p(x^p_t) + F^p_t(x^p_t)x^k_t + G^p_t(x^p_t)v^p_t \]
\[ x^k_{t+1} = f^k(x^p_t) + F^k_t(x^p_t)x^k_t + G^k_t(x^p_t)v^k_t \]
\[ y_t = h(x^p_t) + H_t(x^p_t)x^k_t + e_t, \]
where $F^k_t$, $F^k_p$, $G^p_t$, $G^k_t$ and $H_t$ are linear state transition functions that depend on the non-linear particle states, $x^p_t$, and $f^p$ and $f^k$ are non-linear state transition functions. The measurement function, $h$, is assumed to be non-linear. It is also assumed that the noises are distributed according to

$$
\begin{align*}
    v^p_t &\sim N(0, Q_t^p) \\
    v^k_t &\sim N(0, Q_t^k) \\
    e^k_t &\sim N(0, R_t).
\end{align*}
$$

(3.3)

Like before, the problem is to recursively estimate the posterior probability density function, $p(x_t|y^t_t) = p(x^p_t, x^k_t|y^t_t)$. The same problem can be solved by estimating $p(X^p_t, X^k_t|y^t_t)$ [17], where $X_t = \{x_t\}_{t=0}^T$. To see how this can be solved using Rao-Blackwellization the pdf is rewritten as

$$
p(X^p_t, x^k_t|y^t_t) = p(x^k_t|X^p_t, y^t_t)p(X^p_t|y^t_t).
$$

(3.4)

The pdf is divided into two parts. In [17] it is shown that the first part, $p(x^k_t|X^p_t, y^t_t)$, has the distribution $N(x^k_{t|t}, P^k_{t|t})$, where the recursions for $x^k_{t|t}$ and $P^k_{t|t}$ are given by the Kalman filter

$$
\begin{align*}
    x^k_{t|t} &= \hat{x}^k_{t|t-1} + K_{\text{mu},t}(y_t - h(x^p_t) - H_t\hat{x}^k_{t|t-1}) \\
    P^k_{t|t} &= P^k_{t|t-1} - K_{\text{mu},t}S_{\text{mu},t}K_{\text{mu},t}^T \\
    K_{\text{mu},t} &= P^k_{t|t-1}H_t^T S_{\text{mu},t}^{-1} \\
    S_{\text{mu},t} &= R_t + H_tP^k_{t|t-1}H_t^T.
\end{align*}
$$

(3.5a) - (3.5d)

$$
\begin{align*}
    x^k_{t+1|t} &= F_t^k x^k_t + K_{\text{uu},t}(x^p_{t+1} - f^p(x^p_t) - F_t^k \hat{x}^k_{t|t}) + f^k(x^p_t) \\
    P^k_{t+1|t} &= F_t^k P^k_{t|t}(F_t^k)^T - K_{\text{uu},t}S_{\text{uu},t}K_{\text{uu},t}^T + G^k_t Q^k_t (G^k_t)^T \\
    K_{\text{uu},t} &= F_t^k P^k_{t|t}(F_t^k)^T S_{\text{uu},t}^{-1} \\
    S_{\text{uu},t} &= G^p_t Q^p_t (G^p_t)^T + F_t^k P^k_{t|t}(F_t^k)^T.
\end{align*}
$$

(3.6a) - (3.6d)

The second part, $p(X^p_t|y^t_t)$, can again be rewritten using Bayes rule

$$
p(X^p_t|y^t_t) = \frac{p(y_t|X^p_t, y^t_{t-1})p(x^p_t|x^p_{t-1}, y^t_{t-1})}{p(y_t|y^t_{t-1})}.
$$

(3.7)

This pdf is best solved using a particle filter, since the states $x^p_t$ was assumed to be non-linear. The whole idea with Rao-Blackwellization is to decrease the variance for a given number of particles. Instead of decreasing the variance, the number of particles could be decreased keeping the same variance in order to lower the calculation burden. As can be seen in (3.6), the update of $P^k_{t+1|t}$ depends on $F_t^k$, $F_t^p$, $G^p_t$, $G^k_t$ and in (3.5) the update of $P^k_{t|t}$ depends on $H_t$. If $P$ has to be updated for every particle, the calculation burden will be too high, so a prerequisite has to be that $F_t^p$, $F_t^k$, $G^p_t$, $G^k_t$ and $H_t$ are independent of the particle states, $x^p_t$. 
3.1 Partitioning of the State Space Model

3.1.1 The Gain in Using Rao-Blackwellization

Rao-Blackwellization strives to lower the variance of the error in the calculation of integrals of the form

$$\bar{g}_t = \mathbb{E}_{p(x_t|Y_t)}[g(x_t)] = \int g(x_t)p(x_t|Y_t)\,dx_t \approx \sum_{i=1}^{N} g(x_t^{(i)}) = \bar{g}_t^N, \quad (3.8)$$

where $x_t^{(i)}$ are i.i.d. samples drawn from the probability density $p(x_t|Y_t)$. The variance decrease that is obtained by applying Rao-Blackwellization is given by Lemma 3.2. in [17, p. 48]

$$\sigma_t^2 - \sigma_{rb,t}^2 = \mathbb{E}_{q(X_t|Y_t)}\left[\text{Var}_{q(X_t|Y_t)}\left((g(x_t) - \bar{g}_t)w(X_t)\right)\right] \geq 0. \quad (3.9)$$

In order to gain a lot from Rao-Blackwellization, the variance of $g(x_t)$ has to be significant. The part of the state space model that the Kalman filter operates on is $x_{kt}$, so it is reasonable to believe that this is where the gain is made. Therefore, let $g(x_t) = x_{kt}$, and assume that $H_t(x_{pt}) = 0$ in (3.1) and that the importance function is $q(X_t|Y_t) = p(X_t|Y_{t-1})$. The gain can then be approximated by

$$\sigma_t^2 - \sigma_{rb,t}^2 \approx \sum_{i=1}^{N} P_{t}^{k,(i)}(\bar{w}_t^{(i)})^2. \quad (3.10)$$

To tell if this is a big improvement, this can be compared with the resulting variance

$$\sigma_{rb,t}^2 \approx \sum_{i=1}^{N} P_{t}^{k,(i)}(\bar{x}_t^{(i)})^2. \quad (3.11)$$

This means that if the covariance matrices $P_{t}^{k,(i)}$ are big compared to the corresponding spread of the mean terms, $(\bar{x}_t^{(i)})^2$, there is a lot to gain by using Rao-Blackwellization. Later on, $x_{kt}$ will be the radius difference between the right and the left rear tire of the vehicle and intuitively, this entity will have relatively small variance, so this indicates that unfortunately there might not be much gain to be made by using Rao-Blackwellization in the case of vehicle positioning. For a deeper explanation and derivation of the formulae used above, see [17, pp. 42–52].
Chapter 4

Map Aided Positioning

The estimates that are interesting in the case of vehicle positioning are the absolute position and orientation of the vehicle. A position has to be given in a reference frame. For convenience the same reference frames used in [13] will also be used here, see Figure 4.1. That is, a body fixed reference frame \{\hat{x}_{\text{long}}, \hat{x}_{\text{lat}}\} and a map reference frame \{\hat{x}_1, \hat{x}_2\}, whose coordinates are given as \((x, y)^T\). The orientation of the body fixed frame relative the map frame is given by the vehicle’s angle of direction, \(\psi\).

Let \(\chi(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}\) be the position vector of the vehicle, \(\psi(t)\) the vehicle’s angle of direction at time \(t\). Then the state equations should be expressed as

\[
\begin{align*}
\dot{\chi}(t) &= f(\psi(t), V(u(t))) + v_\chi(t) && (4.1a) \\
\dot{\psi}(t) &= \dot{\psi}(u(t)) + v_\psi(t) && (4.1b) \\
y_{\text{meas}}(t) &= h(\chi(t), \psi(t)) + e(t), && (4.1c)
\end{align*}
\]
where \( V(u(t)) \) is the speed in the longitudinal (heading) direction, \( \dot{\psi}(u(t)) \) is the change of the angle of direction (yaw rate), \( f \) and \( h \) are non-linear functions, \( v_i(t) \) is the process noise for component \( i = \chi, \psi \) and \( e(t) \) is the measurement noise. The speed \( V(u(t)) \) and yaw rate \( \dot{\psi}(u(t)) \) are not directly given as inputs, but they can be calculated from the wheel speeds, \( u(t) \), in various ways. This will be described in detail in Section 4.2. The non-linear function \( f \) is given by the coordinate transformation

\[
\begin{pmatrix}
-V(u(t)) \sin(\psi(t)) \\
V(u(t)) \cos(\psi(t))
\end{pmatrix}.
\]  

(4.2)

The inputs in the case of positioning a vehicle according to a map are mainly the wheel speeds. Extra sensor information from gyros, accelerometers and GPS could also be present in some cases, but the wheel speeds are the basic sensors that should always be present. In order to incorporate the map information, a function \( y_{\text{meas}}(t) = h(\chi(t), \psi(t)) + e(t) \), representing a measurement, is introduced. The measurement noise \( e(t) \) is assumed to be white Gaussian noise and \( h \) is some non-linear function. For instance, \( h(\chi(t), \psi(t)) \) could be chosen to be the minimum Euclidean distance from the point \( \chi(t) \) to the road network \( \Omega_R \), (see Figure 4.2)

\[
h(\chi(t), \psi(t)) = \min_{\chi \in \Omega_R} ||\chi(t) - \chi||_2.
\]  

(4.3)
To take traffic information, such as one-way streets and roundabout rules etc., into account the function could be altered to punish roads that are “unlikely”. For instance, it is highly unlikely that someone would drive the wrong way around a roundabout. The new measurement function could then look like

\[
h(\chi(t), \psi(t)) = \min_{\chi \in \Omega_R} ||\alpha(r(\chi, \psi))(\chi(t) - \chi)||_2,
\]

where \(\alpha(r(\chi, \psi))\) is the penalty function applied to the road \(r(\chi)\) that is associated with the closest point \(\chi\) in the road network \(\Omega_R\).

### 4.1 Obtaining Information From the Map

The map consists of a large collection of roads and each road consists of a number of segments. Each segment is defined by a start and an end node. The road segments are assumed to be straight lines between these nodes.

The particle filter incorporates the information from the map via the measurement equation \(y_t = h(\chi_t, \psi_t) + e_t\). Since the vehicle in the normal case is considered to be on a road, this will be a measure of how far from the road the particles are and how the particles are oriented compared to the road. The most natural measurement is the Euclidean distance to the nearest road segment and the difference between the particles angle of direction and the direction of the road segment, but there are a lot of other measures, such as the Rotational Variation Metric described in [15].

#### 4.1.1 Finding the Closest Road Segment

The most straight forward way to find the closest road segment is to use brute force. This means that for each particle, all the road segments has to be iterated in order to find the closest match. This can be really cumbersome if the number of particles is large and if the map contains a lot of roads. The first thing to do in order to decrease the calculation burden is obviously to reduce the number of roads to search. Since the particles are limited in space, a bounding box can be defined that guarantees that no road that lies entirely outside it is closest to any particle. Actually this only holds if the box contains at least one road, but if it does not the vehicle is said to be off-road anyway.

Now consider a set of \(M\) road segments that lie within the bounding box. The segments are defined by their nodes:

\[
\bar{R} = \{a_{i,x}, a_{i,y}, b_{i,x}, b_{i,y}\}_{i=1}^M
\]

Assume that \(N\) particles are positioned according to:

\[
\bar{P} = \{x_i, y_i\}_{i=1}^N
\]
With these definitions an upper bound of the distance from particle $j$ to the nearest road can be calculated by observing that the nearest road can lie no further away than

$$r_j \leq \min_i \left( \min(|x_j - a_{i,x}| + |y_j - a_{i,y}|, |x_j - b_{i,x}| + |y_j - b_{i,y}|) \right). \quad (4.7)$$

This is a direct result of the triangular inequality theorem where $|x + y| < |x| + |y|$ for all non-parallel vectors $x$ and $y$. The upper bound calculation contains no costly operations, so it should be a quite quick operation. Once this upper bound is calculated a course lower bound can be found for each road:

$$r_{j,i} \geq \max \left( \min(|x_j - a_{i,x}|, |x_j - b_{i,x}|) + \frac{|a_{i,x} - b_{i,x}|}{2}, \min(|y_j - a_{i,y}|, |y_j - b_{i,y}|) + \frac{|a_{i,y} - b_{i,y}|}{2} \right). \quad (4.8)$$

So, if $r_{i,j} \leq r_j$ then the exact minimum distance from particle $P_j$ to road segment $R_i$ should be calculated, otherwise the road is guaranteed not to be the closest road. The exact minimum distance from particle $P_j$ to some point on the road segment $R_i$ is obtained as ([3])

$$r_{j,i} = (ax_j + by_j + c) \quad (4.8a)$$

$$a = \frac{a_{i,y} - b_{i,y}}{d} \quad (4.8b)$$

$$b = \frac{a_{i,x} - b_{i,x}}{d} \quad (4.8c)$$

$$c = \frac{a_{i,x}b_{i,y} - b_{i,x}a_{i,y}}{d} \quad (4.8d)$$

$$d = \sqrt{(a_{i,x} - b_{i,x})^2 + (a_{i,y} - b_{i,y})^2} \quad (4.8e)$$

What is actually wanted in the measurement function $h(\chi_t, \psi_t)$ is the position residue and the angular residue. Therefore, once the closest road has been found, the minimum distance and a pointer to the road and the road segment is stored. This is to be able to calculate the angle of the road to get a measurement of the difference between road angle and heading of each particle. One problem not dealt with in the above reasoning is that the closest road is so far only a measure in the $\{x, y\}$ space, the $\psi$ measure is only considered in the calculation of the directional residues, which affects the weights, not in the minimum distance calculation. The effect is that the closest road might not be that likely, since there could be a big difference in heading angle and road angle. For instance in intersections where roads are equally close, the angle should be considered to separate them [3, 10].

**Road Intersections**

Consider a cloud of particles approaching a road intersection. Due to the artificial noise some particles are spread somewhat off the road on both sides. When these particles reach the intersection, the closest road according to the above definitions
will become the road that intersects the road that the cloud is “travelling” on. It might be true that this actually is the closest road in a strict Euclidean sense, but probabilistically it must be the road that is both close and has about the same direction, because if the road that is orthogonal to the direction of travel is chosen as the closest, these particles will become very unlikely and soon die out, and the cloud will become hour-glass shaped around intersections, see Figure 4.3. To improve the statistical efficiency, in a similar way that the APF algorithm does, the angular discrepancy is included in the distance measure in order to prune out these roads earlier and give chance to the more likely road instead. This would also affect the upper bound calculation, but since the angular difference should be used only to separate the closest roads, the upper bound can be kept as before. The same goes for the lower bound calculation. It is when it comes to the more precise projection of the particles onto roads that the new measure should be used. The major reason for this is to lower the calculation burden, since the angle calculation is quite computationally expensive. The final calculation of the distance should then be replaced with

\[ r_{j,i} = (ax_j + by_j + c) + \beta \min(|\psi_j - \psi_{\text{road}}|, |\psi_j - \psi_{\text{road}} + \pi|) \]  

\[ a = \frac{a_{i,y} - b_{i,y}}{d} \]  

\[ b = \frac{a_{i,x} - b_{i,x}}{d} \]  

\[ c = \frac{a_{i,x}b_{i,y} - b_{i,x}a_{i,y}}{d} \]  

\[ d = \sqrt{(a_{i,x} - b_{i,x})^2 + (a_{i,y} - b_{i,y})^2} \]  

\[ \psi_{\text{road}} = \tan^{-1}(a_{i,x} - b_{i,x}, a_{i,y} - b_{i,y}) \]

where the \( \tan^{-1}(x, y) \) function compensates for which quadrant \( x, y \) is in. Note that the calculation of the angular difference has to consider that a road has two directions so it is the minimum discrepancy that should be added as a punishment.

![Figure 4.3. Particle behavior around road intersections.](image)
Connectability

Since the map is stored as a network, the connectability between the roads could be used to find the closest roads. If a particle is closest to road segment $i$, the closest road in the next measurement will be the same road segment or a road segment connected to this segment. This technique could speed up the calculation considerably. There is one drawback though, the particles can not “jump between roads” which could lead to effects such that some roads are neglected, for instance on roads with multiple intersections. This could be solved by searching the connectability tree further, but this will not guarantee that all roads are found and the computational gain is lost.

A simpler scheme is to first try the road segment that was closest to the particle in the previous iteration. If this road is closer than a certain threshold this road segment is considered to still be the closest, otherwise the linear search method above is performed. Simulations show that the previous closest road still is the closest in $\sim 90\%$ of the cases, thus speeding up the search considerably.

4.1.2 Including Traffic Information

One of the main objectives is to make the filter more robust. One idea to achieve this is to take traffic information into account. The digital map does not only contain coordinates of the road segment of the road network, but also a lot of additional information. Some examples of the information provided are:

- Speed limit.
- Size of the road.
- If the road is a roundabout.
- If the road is one way street.
- Accuracy of the coordinates.

The tool to use to weigh in this kind of information is the distance residue. To make a road, for instance a one way street, less likely it is just to add a punishment to this residue in the same way as the angular residue was added in Section 4.1.1. This will either make another road “closer”, or if the road still is the closest, increase the residue and therefore making particles travelling this road less likely.

4.2 The State Space Model

One objective in this thesis is to improve the MAP algorithm and since the core of the algorithm is the particle filter this is the obvious choice to start improving. There are two sides to the filter. One is the model that the filter operates on, and the second is the actual filter and its parameters. In this section the model will be considered before trying different versions of particle filter algorithms.
The state space model (4.1) above contained the entities $V$ and $\psi$ that describe the speed and yaw rate of the vehicle. Assume that the only given inputs are the wheel speeds. The wheel speeds of interest are those of the non-driven wheels, here assumed to be on the rear axle of the car. Denote these $w_{rr}$ and $w_{rl}$ for rear right and $rl$ for rear left, see Figure 4.4. If the radius of each tire is known ($r_{rr}$ and $r_{rl}$), and no wheel slip is present then the mean speed of the vehicle, i.e. the speed of the middle of the rear axle of the car, can be calculated as

$$V(w_{rr}(t), w_{rl}(t)) = \frac{w_{rr}(t)r_{rr} + w_{rl}(t)r_{rl}}{2}. \quad (4.10)$$

If the track gauge (the distance between the tires), $L$, is known, the yaw rate of the vehicle can be expressed in terms of the wheel speeds. Here the wheel slip might seem important to take into account, but in [8] it is shown that this can be neglected. The expression below is based on the assumption that the vehicle can be modeled as a rigid body

$$\dot{\psi}(w_{rr}(t), w_{rl}(t)) = \frac{w_{rr}(t)r_{rr} - w_{rl}(t)r_{rl}}{L}. \quad (4.11)$$

If these values were known and fixed, the state model in (4.1) would be enough to describe the vehicle with a relatively high degree of accuracy. However, the tire radii can not be considered to be constant in time. Variations due to pressure differences and acceleration effects occur. To compensate for this, the tires are

![Figure 4.4. The rear axle, definitions](image)
modeled to have a nominal radius \( r_n \). Since this radius can vary due to for instance tire pressure, an offset to this radius, \( \delta r^\text{tn} \) is also modeled. Finally, since the tire pressure can be different on each side of the vehicle, an extra offset difference, \( \delta r^\text{tl} \), modeling this will also be added. The distance between the centers of the tires can be regarded constant, but the value of it might not be known exactly. So an offset, \( \delta L \), can be added in the model. The offsets, \( \delta r^\text{tn}, \delta r^\text{tl} \) and \( \delta L \), are assumed to be constant. Their derivatives are therefore modeled as white Gaussian noise, \( v_{\xi(t)} \), with zero mean and a variance reflecting the uncertainty of the initial value of the corresponding entity. This yields the state space model

\[
\begin{align*}
\dot{x}(t) &= -V(w_{rr}(t), w_{rt}(t))\sin(\psi(t)) + v_x(t) \\
\dot{y}(t) &= V(w_{rr}(t), w_{rt}(t))\cos(\psi(t)) + v_y(t) \\
\dot{\psi}(t) &= \dot{\psi}(w_{rr}(t), w_{rt}(t)) + v_\psi(t) \\
\dot{\delta}^\text{tn}(t) &= v_y(t) \quad (4.12) \\
\dot{\delta}^\text{tl}(t) &= v_y(t) \\
\dot{\delta}^L(t) &= v_y(t) \\
y_{\text{meas}}(t) &= h(x(t), y(t), \psi(t)) + \epsilon_t,
\end{align*}
\]

where

\[
V(w_{rr}(t), w_{rt}(t)) = \frac{w_{rr}(t)(r_n + \delta r^\text{tn}(t) - \delta r^\text{tl}(t)) + w_{rt}(t)(r_n + \delta r^\text{tn}(t) + \delta r^\text{tl}(t))}{2} - \frac{w_{rr}(t) - w_{rt}(t)}{2}\delta^\text{rt}(t) \quad (4.13)
\]

and

\[
\dot{\psi}(w_{rr}(t), w_{rt}(t)) = \frac{w_{rr}(t)(r_n + \delta r^\text{tn}(t) - \delta r^\text{tl}(t)) - w_{rt}(t)(r_n + \delta r^\text{tn}(t) + \delta r^\text{tl}(t))}{L + \delta^\text{rt}(t)} - \frac{w_{rr}(t) - w_{rt}(t)}{L + \delta^\text{rt}(t)}\delta^\text{rt}(t) \quad (4.14)
\]

There are some things to observe in this state space model. First of all, to clarify, the artificial measurement containing the map information, \( y_{\text{meas}}(t) \), is not a scalar but contains measurements of both position and direction. Another remark is that the longitudinal drift \( \delta r^\text{tn} \) aims at estimating the offset in the tire radius, but since the map is flat, and the world is not, the inclination of the road will affect this drift. In slopes, this estimate will be larger than the actual offset, so this entity might as well be modeled as some extra noise in the speed of the vehicle. Also the difference offset \( \delta r^\text{tl} \) will be affected during cornering since the vehicle is accelerated and the forces on the tires will alter their radii during the turn.

### 4.2.1 Discretization

The wheel speeds are sampled regularly, with sample time \( T_s \), so a time discrete model is needed. Assumptions made in the discretization process are that the noise \( v_{\xi(t)} \) is piecewise constant and that the wheel speeds and the offsets \( \delta r^\text{tn} \) and \( \delta r^\text{tl} \)
are piecewise linear. To make the discrete state space model a bit less complex, the offset \( \delta^L \) and the yaw rate \( \psi \) are considered to be piecewise constant. Also, the discrete time index \( t + 1 \) stands for \( t + T \) to make the notation more convenient.

The discretized version of (4.12) becomes

\[
x_{t+1} = x_t - T_s V \left( \frac{w_{rr,t} + w_{rr,t+1}}{2}, \frac{w_{rl,t} + w_{rl,t+1}}{2} \right) \sin(\psi_t) + v_{x,t} -
\]
\[- \frac{T_s^2}{4} \left( w_{rr,t} + w_{rr,t+1} + \frac{w_{rl,t} + w_{rl,t+1}}{2} \right) \sin(\psi_t) \]
\[
y_{t+1} = y_t + T_s V \left( \frac{w_{rr,t} + w_{rr,t+1}}{2}, \frac{w_{rl,t} + w_{rl,t+1}}{2} \right) \cos(\psi_t) + v_{y,t} +
\]
\[- \frac{T_s^2}{4} \left( w_{rr,t} + w_{rr,t+1} + \frac{w_{rl,t} + w_{rl,t+1}}{2} \right) \cos(\psi_t) \]
\[
y_{t+1} = \psi_{t+1} = \psi_t + T_s \psi \left( \frac{w_{rr,t} + w_{rr,t+1}}{2}, \frac{w_{rl,t} + w_{rl,t+1}}{2} \right) + v_{\psi,t} +
\]
\[- \frac{T_s^2}{4} \left( w_{rr,t} + w_{rr,t+1} + \frac{w_{rl,t} + w_{rl,t+1}}{2} \right) \left( v_{\delta^r,t} - v_{\delta^l,t} \right) \]
\[
d_{\delta^r,t+1} = d_{\delta^r,t} + v_{\delta^r,t} \]
\[
d_{\delta^l,t+1} = d_{\delta^l,t} + v_{\delta^l,t} \]
\[
y_{\text{meas},t} = h(x_t, y_t, \psi_t) + e_t, \]

(4.15)

where \( V \) and \( \dot{\psi} \) are defined as in (4.13) and (4.14). The assumption that the wheel speeds are piecewise linear makes it necessary to wait for an extra measurement of \( w_{rr}(t+1) \) and \( w_{rl}(t+1) \). To simplify the state space model above further, this assumption can be replaced with the assumption of piecewise constant wheels speeds. This gives the following

\[
x_{t+1} = x_t - T_s \left( \frac{w_{rr,t} + w_{rr,t+1}}{2}, \frac{w_{rl,t} + w_{rl,t+1}}{2} \right) \sin(\psi_t) -
\]
\[- \frac{T_s^2}{4} \left( w_{rr,t} + w_{rr,t+1} + \frac{w_{rl,t} + w_{rl,t+1}}{2} \right) \sin(\psi_t) \]
\[
y_{t+1} = y_t + T_s \left( \frac{w_{rr,t} + w_{rr,t+1}}{2}, \frac{w_{rl,t} + w_{rl,t+1}}{2} \right) \cos(\psi_t) + v_{y,t} +
\]
\[- \frac{T_s^2}{4} \left( w_{rr,t} + w_{rr,t+1} + \frac{w_{rl,t} + w_{rl,t+1}}{2} \right) \cos(\psi_t) \]
\[
y_{t+1} = \psi_{t+1} = \psi_t + T_s \psi \left( \frac{w_{rr,t} + w_{rr,t+1}}{2}, \frac{w_{rl,t} + w_{rl,t+1}}{2} \right) + v_{\psi,t} +
\]
\[- \frac{T_s^2}{4} \left( w_{rr,t} + w_{rr,t+1} + \frac{w_{rl,t} + w_{rl,t+1}}{2} \right) \left( v_{\delta^r,t} - v_{\delta^l,t} \right) \]
\[
d_{\delta^r,t+1} = d_{\delta^r,t} + v_{\delta^r,t} \]
\[
d_{\delta^l,t+1} = d_{\delta^l,t} + v_{\delta^l,t} \]
\[
y_{\text{meas},t} = h(x_t, y_t, \psi_t) + e_t, \]

(4.16)
4.2.2 Noise Models

An important aspect of the modeling is how to chose the noise model. The noise for the drifts, $\delta_{rn}, \delta_{rt}$ and $\delta_L$ are chosen as white Gaussian noise for simplicity. Other distributions have been tried but no major difference could be observed. The design parameter here is the variance of the noise. A large variance means a high uncertainty in the parameter, but allows faster variations. The noise in the drifts are propagated to both the positional and directional noise via the speed and yaw rate calculations, see (4.13) and (4.14). For instance, the longitudinal noise, $\delta_{rn}$, that models an offset in the nominal tire radius, $r_n$, will spread the particles along the direction of travel.

The noise in the direction, $v_\psi$, is propagated to positional noise due to the geometric transformation (4.2). This means that a change in the directional noise affects the noise in the position as well as the direction, see Figure 4.5.

The positional noise, $v_x,v_y$, compensates for the depletion problem caused by the resampling step in the algorithm. This noise can also be correlated so that the particles are spread more in the direction of travel. This gives the same effect as if noise were added to the speed, $V(u_t)$. If the longitudinal drift is modeled, i.e. $\delta_{rn}$ is part of the model, the correlation of the positional noise is not necessary. In fact, it should not be correlated since this could corrupt the statistical properties for the estimation of the longitudinal drift.

![Figure 4.5](image-url)

**Figure 4.5.** Different noise settings for the directional noise, $v_\psi$. The left figure shows a particle cloud that is spread using a directional noise with a standard deviation of 2 degrees. The right figure shows a particle cloud that is spread using a directional noise with a standard deviation of 12 degrees.
4.2.3 Observability

As previously mentioned, the longitudinal drift $\delta^r$, is affected by the inclination of the road. There is also very little information about the longitudinal drift on long straight roads. This suggests that this drift might not be fully observable. The problem with (4.12) is that the measurement function $h(x(t), y(t), \psi(t))$ is highly non-linear and even impossible to write as a closed form expression, so checking the observability is not trivial.

The estimation of the length of the rear axle is based on the fact that a vehicle with a long axle turns slower than one with a short axle. If the axle length is incorrect the rate of turn will be incorrect and therefore also the angle. To be able to estimate the length of the axle the vehicle has to turn to get a difference in angle that can be compared to the angle of the road. This means that the $\delta^L$ estimate is only observable when turning, $w_{rr} \neq w_{rl}$, and the accuracy is dependent on the accuracy of the map.

The estimation of the tire radius difference between the right and left tire gives rise to a lateral drift, so this estimate should be observable on straight roads because an incorrect radius offset, would make the particles turn, and if the road is straight, these particles will be unlikely and eventually filtered out.

If MAP is run as a complement to a GPS system, the tire radius offset, $\delta^r$, should be observable since the GPS provides a measurement that gives information on straight roads as well.

4.2.4 Evaluation of the State Space Model

In order to actually find out if the model is observable given the measurements $y_{\text{meas}, t} = h(x_t, y_t, \psi_t)$, the model has to be tested. If the estimates converge, they can be regarded as observable. If they do not, either they are not observable or the model is incorrect.

The test-runs were performed using a standard SIS particle filter in 100 run Monte Carlo simulations. In the simulations, the axle width, $L$, and the nominal tire radius, $r_n$, was set to a few different fixed values. The offset estimates, $\delta_{rn}$ and $\delta_L$, are estimates of the difference between the set fixed value and the “true” value. The true value was obtained by tuning the parameter so that the dead reckoned trajectory matched the map\(^1\).

\(^1\)The test-run used to evaluate the rear axle width offset, $\delta^L$ and the offset in nominal tire radius, $\delta^r$, is shown in Figure C.1 in Appendix C
Width of the Rear Axle

The estimate for $\delta^L$ seems to contain some information about the error in axle width, but the estimates has some sharp changes at discrete points. The reason is that the information about $\delta^L$ is only provided when the vehicle turns, it is partly observable. Figure 4.6 shows the mean estimate of the 100 runs for three different axle widths and the corresponding standard deviations. The results indicate that it is possible to estimate the width of the rear axle. Despite this, the choice to regard the axle width, $L$, as known has been made. This is motivated by the fact that the axle width really is constant in time and the value is easily obtained, either from a manual during the installation of MAP or from the CAN bus, and by the fact that adding the axle width offset as an extra state increases the complexity of the problem making it more computationally expensive.

Figure 4.6. Estimation of rear axle width offset and its standard deviation for three different axle width settings.
Nominal Tire Radius

As with the estimate of the axle width offset, a good estimate for $\delta_r^m$ is hard to find, when no GPS signal is available. In Figure 4.7 it is clear that the information lies in road turns. Each “edge” in the estimate corresponds to a corner in the road. The estimate becomes more certain at these discrete points (the standard deviation decreases), but the uncertainty increases between the turns. Another thing to note is that an increase in diverged runs occurred when trying to estimate this longitudinal drift. In many cases the runs diverged because of minor map errors, that resulted in an incorrect estimate. The diverged runs were removed in the calculation of the mean estimate presented in the figure.

![Graph showing estimation of tire radius offset and its standard deviation for three different tire radius settings.](image)

**Figure 4.7.** Estimation of tire radius offset and its standard deviation for three different tire radius settings.
Estimating the longitudinal drift is very appealing, since this would improve off-road performance considerably and also decrease the risk for divergence after long straight roads. Therefore, a lot of effort has been put into this problem, but no solution has been found.

There is no observability problem with the longitudinal drift, $\delta_{rn}$, when MAP is run together with GPS. Figure 4.8 shows the mean offset estimate and its standard deviation, for three different values on the nominal tire radius. This shows that if MAP is run as a complement to GPS, MAP will have very good estimates for both lateral and longitudinal drifts and this is very useful if the GPS fails.

![Figure 4.8](image_url)

**Figure 4.8.** Estimation of tire radius offsets and their standard deviation when the algorithm is exploiting the GPS signal.
Tire Radius Difference

In the stand-alone configuration, without GPS support, the only parameter, apart from the position and direction, that seems to be worth estimating is the difference between the right and left tire radius, $\delta^t_l$. This estimate shows good observability results, see Figure 4.9.

![Figure 4.9. Estimate (solid) of tire radius difference and standard deviation (dotted).](image)

The result also shows that the offset varies when the vehicle turns. When turning, the vehicle gets a lateral acceleration that is taken up by the tires. This results in different forces and therefore different tire radii on both sides of the vehicle. Attempts have been made to compensate for this effect in order to get a more stable estimate. To do this, the tire is modeled as an ideal spring. The new model for the offset $\delta^t_l$ becomes

$$
\delta^t_l = -\gamma \frac{\dot{\psi}_t(w_{rr,t}, w_{rl,t}) w_t(w_{rr,t}, w_{rl,t})}{L + \delta^t_L} + \delta^t_L
$$

(4.17a)

$$
\gamma = \frac{2m h_{CoG}}{k_{tire}},
$$

(4.17b)

where $m$ is the mass of the car, $h_{CoG}$ is the height to the vehicle’s center of gravity and $k_{tire}$ is the spring constant of the tire. However, the compensation for the

\footnote{For derivation see Appendix A}
acceleration when turning gives poor or no improvements at all. The reason for this can be a too simple tire model or that the inputs are too uncertain. When looking at the acceleration that affects the car\(^3\), it is clear that the signal is too noisy as it is. After applying a low pass filter, the signal looks quite a lot like the estimate. This means that the filtered signal could be subtracted from the estimate to get rid of some of the acceleration effects\(^4\). Another issue is that there are a lot of unknown parameters, that in this case has been tuned off-line to give a good result. Some of these parameters, such as \(m\) and \(h_{\text{CoG}}\), are not constant and if they change the model will be corrupted. Another way to get a relatively good estimate is to use the mean estimate instead of the actual estimate via some form of filtering, for instance a forgetting factor filter.

Estimating the drifts are not that important for the filter to work properly, since extra noise is added in order to excite as many modes as possible. The problem comes when the vehicle drives off-road. Then there is no map information to calibrate on. This makes long off-road drives hard to position in a precise way.

### The Resulting State Model

Since neither of \(\delta^L\) and \(\delta^r\) can be estimated in a meaningful way, the resulting state model that will be further explored is

\[
x_{t+1} = x_t - T_s \left( \frac{w_{rr,t} + w_{rl,t}}{2} \right) r_n - \frac{w_{rr,t} - w_{rl,t}}{2} \left( \delta_{t} + \frac{T_s}{2} v_{\psi,t} \right) \sin(\psi_t) + v_{x,t} \quad (4.18a)
\]

\[
y_{t+1} = y_t + T_s \left( \frac{w_{rr,t} + w_{rl,t}}{2} \right) r_n - \frac{w_{rr,t} - w_{rl,t}}{2} \left( \delta_{t} + \frac{T_s}{2} v_{\psi,t} \right) \cos(\psi_t) + v_{y,t} \quad (4.18b)
\]

\[
\psi_{t+1} = \psi_t + T_s \left( \frac{w_{rr,t} - w_{rl,t}}{L} \right) \left( \frac{T_s}{2} v_{\psi,t} \right) - \frac{w_{rr,t} + w_{rl,t}}{L} \delta_{t} + v_{\psi,t} \quad (4.18c)
\]

\[
\delta_{t+1} = \delta_{t} + v_{\delta,t} \quad (4.18d)
\]

\[
y_{\text{meas},t} = h(x_t, y_t, \psi_t) + e_t. \quad (4.18e)
\]

### Evaluating the Various Filters

The new approaches in this thesis are the Auxiliary Particle filter and the Rao-Blackwellized filter. Both filters aim at lowering the variance of the estimation error which can be used to lower the number of particles or to increase the artificial noise and therefore also increasing the robustness.

#### 4.3.1 The Auxiliary Particle Filter

The implementation of the APF was done using the algorithm in Section 2.2.3. To evaluate the effect of using an APF instead of a standard SIS filter Monte Carlo simulations using the different implementations have been performed. To assure that the tests cover most of the properties of the filters, the simulations have been

\(^3\)See Figure B.1 in Appendix B

\(^4\)See Figure B.2 in Appendix B
made using a few different test drives and a number of different parameter settings. The most interesting criterion is of course the magnitude of the absolute error in the estimated position. If the measurements are informative, the APF should provide a better performance than the ordinary particle filter. The results show that the APF had a slightly lower variance in some of the test runs (see Figure 4.11), but the difference is not significant. The APF is supposed to perform best when the measurements are informative. However if the map is a bit erroneous, the APF can even perform worse than the regular particle filter. It was observed in the simulations that although the variance of the position was a bit lower for the APF, the variance for the $\delta^T$ estimate was a lot higher. The reason for this is that the measurements are uncertain due to the discretization of the map. It can also be noted that out of 100 Monte Carlo runs, 2 of the runs took the wrong track in the APF case, whereas the SIS filter converged in all 100 runs.

![Figure 4.10. Auxiliary particle filter vs. SIS filter](image-url)
4.3.2 The Rao-Blackwellized Filter

To perform Rao-Blackwellization on the particle filter, the state space model (4.18) can be partitioned into a non-linear part and a linear part:

\[
x_{t+1}^p = \begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \psi_{t+1} \end{pmatrix} = \begin{pmatrix} x_t - T_s \left( \frac{w_{rr,t} + w_{rl,t}}{2} \right) r_n \sin(\psi_t) + v_{x,t} \\ y_t + T_s \left( \frac{w_{rr,t} - w_{rl,t}}{2} \right) r_n \cos(\psi_t) + v_{y,t} \\ \psi_t + T_s \left( \frac{w_{rr,t} - w_{rl,t}}{L} \right) r_n - \frac{w_{rr,t} + w_{rl,t}}{2} \delta_{rl,t}^{x,i} \sin(\psi_t) + v_{\psi,t} \end{pmatrix}
\]

\[
x_{t+1}^k = \begin{pmatrix} \delta_{t+1}^{x,i} \\ \nu_{\text{mean},t} \end{pmatrix} = \begin{pmatrix} \delta_{t}^{x,i} + v_{\delta^{x,i},t} \\ h(x_t, y_t, \psi_t) + e_t \end{pmatrix}
\]

This model can be rewritten on the same form as (3.2) in Chapter 3.

\[
x_{t+1}^p = \begin{pmatrix} x_{t+1} \\ y_{t+1} \\ \psi_{t+1} \end{pmatrix} = \begin{pmatrix} x_t \\ y_t \\ \psi_t \end{pmatrix} + \begin{pmatrix} -T_s \frac{w_{rr,t} + w_{rl,t}}{2} r_n \sin(\psi_t) \\ T_s \frac{w_{rr,t} - w_{rl,t}}{2} r_n \cos(\psi_t) \\ T_s \frac{w_{rr,t} - w_{rl,t}}{L} r_n - \frac{w_{rr,t} + w_{rl,t}}{2} \delta_{rl,t}^{x,i} \sin(\psi_t) \end{pmatrix} + \begin{pmatrix} v_{x,t} \\ v_{y,t} \\ v_{\psi,t} \end{pmatrix}
\]

\[
x_{t+1}^k = \begin{pmatrix} \delta_{t+1}^{x,i} \\ \nu_{\text{mean},t} \end{pmatrix} = \begin{pmatrix} \delta_{t}^{x,i} + v_{\delta^{x,i},t} \\ h(x_t^p) + e_t \end{pmatrix}
\]

The Rao-Blackwellized, RB, filter was implemented using a standard Kalman filter to solve the analytical part and a SIS filter to handle the non-linear part. The problem here is that the update of the covariance matrix (scalar in this case) \( P \) depends on \( F_t^k(x_t^p) \), see (4.20a), that depends on the particle states, \( x_t^p \). This means that \( P \) needs to be updated separately for each particle, see Equation 3.6, and the calculation burden for the Kalman part will be considerable. This might not be a problem if the gain in robustness is large enough.

100 Monte Carlo runs were performed using a standard SIS filter and the Rao Blackwellized filter presented previously. The results show that the variance of the \( \delta^{x,i} \) estimate was a lot lower for the RB-filter, see Figure 4.11\(^5\). It was noted during the simulations that the choice of the noise parameters for the Kalman filter, \( \{ Q, R \} \), was crucial to get good results. A large \( Q \) (process noise) leads to a fast filter, but with larger variance. Better results was obtained when \( R \) (the measurement noise) was chosen to be larger than \( Q \). The filter became a lot slower, but the variance decreased drastically. Since the parameter to estimate, the tire radius difference, should vary quite slowly, the low speed of the filter is acceptable. These results agree with the theory presented in [12]. All runs converged in this test,

\(^5\)The test-run for comparing RB and SIS is not the same test-run as was run when comparing APF with SIS. The figures should not be used as a comparison between APF and RB.
so to really see the difference in robustness between the two filters another test was performed. In this test the number of particles were varied and the number of diverged runs were counted for the 100 Monte Carlo runs performed for each filter and filter size. The same parameters were used for both filters to be able to compare. The results are shown in Table 4.1.

**Table 4.1.** Number of particles and the corresponding robustness in terms of converged runs for the two filters.

<table>
<thead>
<tr>
<th>#particles</th>
<th>#converged runs SIS</th>
<th>#converged runs RB</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>37</td>
<td>53</td>
</tr>
<tr>
<td>400</td>
<td>72</td>
<td>76</td>
</tr>
<tr>
<td>600</td>
<td>69</td>
<td>81</td>
</tr>
<tr>
<td>1300</td>
<td>76</td>
<td>98</td>
</tr>
<tr>
<td>1500</td>
<td>95</td>
<td>100</td>
</tr>
<tr>
<td>1700</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
From Table 4.1 it is clear that the Rao-Blackwellized filter performs better than the ordinary particle filter. However, since the error covariance matrix needs to be updated separately for each particle, the computational burden for the RB-filter is higher than the gain in robustness.

4.3.3 Tuning

During the evaluation of the particle filters, an insight that has been made is that the filters are sensitive to the choice of noise parameters. Therefore some rules of thumb on how to choose parameters have been developed. The parameters in question are the variances of the measurement noise and the process noise

$$\begin{align*}
\text{measurement noise} & : \begin{bmatrix}
\text{Var}(e^x_t) \\
\text{Var}(e^y_t) \\
\text{Var}(e^{\psi}_{\text{GPS}})
\end{bmatrix} \\
\text{process noise} & : \begin{bmatrix}
\text{Var}(v^x_t) \\
\text{Var}(v^y_t) \\
\text{Var}(v^\psi_t) \\
\text{Var}(v^\delta_{t1})
\end{bmatrix}
\end{align*}$$

(4.21)

The process noise controls how “smeared” out the particles will be and the measurement noise controls how much to trust the measurement or how forgiving the filter will be to measurement errors. To get a grip on what noises to start out with before fine tuning, each individual noise must be put in its context.

The standard deviation for the lateral and longitudinal process noise is chosen to be in the magnitude of a couple of metres. This is to compensate for inclination effects and to lessen the clustering. The longitudinal noise is chosen to be higher than the lateral to spread the particles along the road. The standard deviation for $\psi$ should have a value of a few degrees, to compensate for errors in axle width and noise in the wheel speed sensors. To get a decent estimate, $v^\delta_{t1}$ should have a small standard deviation, in the magnitude of 0.1 mm.

When setting the measurement noise parameters, the parameters should have about the same magnitudes as the corresponding process noise. The variance for the GPS measurement is used to fuse the GPS information together with the particle filters, provided that GPS information is available. The magnitude of this standard deviation can be used to rely more or less on GPS. A basic setting could be a standard deviation of 50 metres.
4.3 Evaluating the Various Filters

Fine Tuning the Process Noise

To see the effects of the various parameters, they have been changed one at a time during a test drive. The parameters can be set differently for the initial convergence phase and the phase after convergence.

The results for the angular variance show that an increase in the $\psi$-variance gives a larger spread in the angle of the particles. This leads to lesser clustering (a more spread out particle cloud). It also gives the effect that more “side tracks” are taken. The increase of this variance leads to a slower convergence. It could give a more robust filter if the map information is poor, but a too large increase in noise gives an unstable filter, since the information in the measurements will be buried in process noise.

As with the angular variance, an increase of the longitudinal and lateral variances also leads to less clustering of the particles. An increase in the lateral variance $\text{Var}(v_y^t)$ will spread the particles “sideways” from the road, whereas an increase in the longitudinal variance $\text{Var}(v_x^t)$ will spread the particles along the road. A too high lateral variance will only increase the sampling ratio since a lot of particles will be highly unlikely. A too high longitudinal variance will cause problem on long straight roads because the particle cloud will be very elongated and the longitudinal uncertainty will become high.

An increase of the variance for $v_\delta^t$ will make the particles take more side tracks since the tire radius difference makes the vehicle turn. Because there are no direct measurement of the tire radius difference a too high a variance will make the $\delta$-estimate interfere with the $\psi$-estimate and the filter will be less stable. On the other hand, a too low variance will make the estimate less sensitive to changes in the tire radiuses which also can make the particles go off track.

Fine Tuning the Measurement Noise

As mentioned before the measurement noise controls how tolerant the filter will be to measurement errors. The goal is to have a robust filter that does not lose track. Since there are map errors the filter must be able to handle these to some extent. Of course, if the map errors are too large, this will have to be handled as off-road driving (see Section 5.1). To handle these errors the process noise can be made large enough to spread the particles on the right track, or the measurements can be forgiving. The latter way has proved itself to be more robust, since the trajectories become more stable and more reliable with less process noise. To get a fast convergence, keep the process noise small and let the measurement be relatively forgiving (large) to maintain a good robustness. To get a good robustness after convergence, keep the process noise smaller than the measurement noise.
Chapter 5

Filter Robustness and Off-road Driving

One goal in this thesis is to make the MAP algorithm more robust. In this case robustness is either of the type that the initial search is robust or of the type that the algorithm can handle imperfections or even missing data in the map. One example of this is when a road does not exist on the map, here called off-road driving.

5.1 Off-road Detection

Before handling the off-road driving itself, the algorithm must first in some way detect that the vehicle has gone off the road or that the map does not correlate with the reality. It is not always easy to do this in an unambiguous way. To be able to grasp what characterizes a particle swarm that is about to go off the road network of the map, two typical cases are considered.

- The road ends dead ahead:
  The swarm of particles is unwilling to leave the road because of the penalty put on it for doing so. Those particles that are furthest from the end of the road is highly likely to be resampled and the swarm will be concentrated to the end of the road. Therefore the variance of the particle cloud will decrease drastically, but the residues will increase because many particles are spread off the road and are therefore resampled.

- The vehicle turns sharply onto a road that is not on the map:
  The particle cloud will first be drawn to the edge of the road in the direction of the turn. Since the angle residues are not considered during sharp turns the particles are likely to survive at first. When the vehicle has completed the turn, the angle residues will be large and most of the particles will be very unlikely. This can be discovered by observing the normalizing factor
(the factor used to normalize the importance weights, $\sum_i w_t^{(i)}$) that in some sense measures degeneration of the particles. A low normalizing factor indicates a high degree of degeneration, that is only a few or no particles are likely. A change in the mean of the normalizing factor can be found using a CUSUM test, which in short is a cumulative sum of the previous factors that exceed a certain threshold. A change in the normalizing factor will make the cumulative sum grow and this can be used as a change detection, see [12, pp. 333–335] for more information on the CUSUM test. By surveilling the $\psi$-residue directly the sharp turn can also be detected. A sharp turn will always yield a large $\psi$-residue, even when the road that the vehicle turns onto is on the map. The difference is that the $\psi$-residue will be restored in the latter case.

CUSUM Test

1. Calculate the normalizing factor as: $f_{\text{norm}},t = -\log(\sum_{i=1}^{N} w_t^{(i)})$, note that $0 \leq f_{\text{norm}},t < \infty$.
2. Calculate the cumulative sum as: $C_{\text{cus}},t = \max(C_{\text{cus}},t-1 + f_{\text{norm}},t - \nu, 0)$.
3. CUSUM alarm if $C_{\text{cusum}} > Q_{\text{alarm}}$

The alarm level, $Q_{\text{alarm}}$, is a design parameter that controls how fast a change is detected. The CUSUM test detects if the mean of the signal, in this case the normalizing factor, $f_{\text{norm}},t$, exceeds the limit set by $\nu$.

5.2 Dead Reckoning

With a certain starting point, the next position of the vehicle can be calculated using dead reckoning\(^1\). Dead reckoning means that by knowing the wheel speeds the speed of the vehicle and rate of turn can be calculated and therefore the next position can be estimated. Assuming that dead reckoning introduces relatively small errors then the vehicle’s position can be estimated with quite good precision. This might seem like taking a step back, since the whole idea with the particle filter was to use the information from the map, but by keeping an extra dead reckoning position, map errors can be easily detected by comparing the filter estimate with the dead reckoning estimate. Dead reckoning needs a known starting position and the quality on this initial position is crucial to get good results. The question is how to get a good starting position. A natural choice is of course to take the filter estimate after filter convergence. The problem is that dead reckoning suffers from

\(^1\)The expression dead reckoning originates from deduced recognition, that is shortened to ded. rec. and misinterpreted as dead reckoning.
5.2 Dead Reckoning

Drifts, so after a while the error in the dead reckoning position will “drift” away and the position will have to be restored. If the filter estimate is chosen as calibrating position it could happen that the vehicle has just gone off the road and then the dead reckoning position will be restored to an “incorrect” value.

5.2.1 Several Estimates of the Dead Reckoning Position

The solution to this quandary is given by keeping several parallel dead reckoning positions that are restored on a regular basis, but not at the same time. By comparing the average discrepancy (including angular difference) between filter estimate and dead reckoning estimate a measure of how good the map information is obtained (see Figure 5.1). This measure can also be seen as an indication of that the vehicle has gone off-road. Another advantage with this technique is that when off-road driving is detected there is already a good estimate of the position in one of the estimates. This way it is possible to “step back in time” to choose the estimate that was initialized just before the error arose and therefore get the best estimate. Furthermore the structure for continuing off-road by dead reckoning is already there. It is just to not restore the estimate and the actual off-road drive can be positioned as well. How to optimally choose the best dead reckoning position has to be further explored. By looking at the various estimates, it is easy to see that at least one of the dead reckoning estimates is really good by comparing it to the GPS position. How to choose this estimate when no GPS support is available, is something that is important to solve to obtain good results during off-road. Simulations show that stepping about five times in the dead reckoning estimates gives relatively good results.

![Figure 5.1. Several dead reckoning estimates](image-url)
5.2.2 Summary on Detection

In the previous section some ways to detect map errors or off-road driving where presented. To sum up, the main entities to observe in order to detect off-road driving are:

- The dead reckoning position residue, that is the average distance from the dead reckoning positions to the filter estimate.
- The dead reckoning direction residue, that is the average difference between the dead reckoning directions and the filter estimate of the direction.
- The normalizing factor in a CUSUM test
- The position residue, that is the average distance from the particles to the nearest road.
- The direction residue, that is the average angular difference between particle direction and the direction of the nearest road.

To be able to detect off-road driving, limits for these values has to be set and to find good limits, various test cases were run. The idea was to find the limits by plotting the residues and classify the patterns into off-road/on-road, see Figures 5.4 and 5.5 at the end of this chapter. In the first case, when the vehicle turns off the road, there is a clear peak in all of the residues. The position residues peak after about 200 samples and that is because of some map imperfections in a roundabout.

In the second case, when the vehicle drives off the road straight ahead, the dead reckoning residues signal off-road. None of the direction residues signal off-road, but this is quite natural since the vehicle drives straight ahead. It should be noted that the direction residue, (upper right figures), is noisy and relatively high. Most of the peaks are due to road intersections which make the residue higher than it really is, see Section 4.1.1.

The resulting detection mechanism is a combination of the different residuals in order not to miss any off-road cases, although it is impossible to guarantee perfect detection results in all conditions. The simulations show that the actual off-road drives are quite easy to detect but false alarms for minor map deficiencies sometimes occur. False alarms should not be a big problem if only the off-road drives are handled in a proper way. If the off-road alarm is false, the off-road handling routine should have no problem finding the road again.

5.3 Finding the Road Again

Once the vehicle has gone off-road, it is easy to just update the dead reckoning position. Since no map information can be used to improve this estimate, this is the best estimate that can be achieved, assuming that the drifts have been estimated
as good as possible before. However, the off-road drive will eventually end and therefore a strategy to find the road that the vehicle drives onto again has to be made.

An observation is that if the off-road drive is short, the error in the dead reckoning position is likely to be quite small, but the longer the off-road drive lasts, the larger the error will be due to drifts. This implies that the “search area” in which to look for possible roads to be back on must expand with time, see Figure 5.2.

Figure 5.2. Uncertainty in position during off-road.

One way to achieve this is to spread the particles as

\[ x^i_t \sim N(\hat{x}^d_t, \sigma(t)), \quad i = 1, \ldots, N \]
\[ \sigma(t) = \sigma_0 + a t, \quad a_{j,k} \geq 0, \quad k = 1, \ldots, M, \quad j = 1, \ldots, M. \]  
(5.1)

Note that \( x_t \) here is the entire state space model and that \( \sigma(t) \) is a matrix describing the uncertainties in the states. The matrix \( a \), with components \( a_{j,k} \), models the increased spreading of the particles over time.

Another way of spreading the particles is to let the difference between the filter estimate and the dead reckoning position be a measurement. Thus, this will effect the calculation of the weights for the particle filter. They are calculated as

\[ w(x) = p(y_t | x_t) = p_{e_t}(y_t - h(x_t)) \]
\[ p_{e_t}(y_t - h(x_t)) \in N(0, \sigma), \]  
(5.2)

where the Gaussian distribution is expanded with two extra dimensions, one for the position residue and one for the angular residue. To trust the dead reckoned position more and rely less on the particle filter, the variance of the residues for the dead reckoning is decreased and the variance for the map residues are increased. This will cause the particle cloud to be “drawn” to the dead reckoned estimate.

It is harder to spread the particles more with time this way, but by adding some extra noise to the position during off-road driving, this will be taken care of.
To detect that the vehicle is back on a road again, the weights of the particles are used. If the particle cloud is close to a road, the particles will have higher weights than if they are far away from a road. Therefore the normalizing factor, \( \sum_{i=1}^{N} w_i^{(t)} \), is a good measure of how near a road the particle swarm is. For off-road detection a CUSUM test was performed on the normalizing factor. The same method can be used for detecting that the vehicle is back on the road. Another way is to simply calculate the minimum distance to the nearest road. If this distance is less than a predefined threshold, for instance 5 m, the vehicle is said to be back on the road.

5.3.1 Off-road When Using GPS Support

This scenario is quite simple. The same mechanisms can be used to detect off-road driving. Also the difference between GPS and MAP estimate could be used for detection, but uncertainty in the GPS signal can cause unwanted results, so this has not been used.

During the off-road drive, the particles are spread around the GPS-estimate, provided that the GPS-integrity is high. They are updated in time using the wheel speeds and the estimated drifts. The same mechanism as was used in the previous section can be used for detecting that the vehicle is back on the road again.

5.3.2 Results

The best result was obtained when using the scheme depicted in Figure 5.3. To detect off-road cases, the position and direction residues are used to give a warning, since these indicate off-road a little earlier than the dead reckoning residues. The dead reckoning residues are used to verify that the warning was not a false alarm. The dead reckoning residues are also used as a stand-alone detection system, with a higher threshold if no warning is present.

Once off-road driving is detected the particles are spread, using a normal distribution, around one of the dead reckoning estimates. The particles are then updated as before using the wheel speed signals, only they are not resampled and their weights are not updated. Since the particles are spread around the dead reckoning position, with slightly different directions, they will form a cloud that will spread out as they are updated. To spread the cloud even more, some extra noise can be added but this tends to make the off-road estimate less precise. The more the particles are spread, the shorter off-road drives can be handled, since the expanding cloud eventually will come across a road.

When the vehicle has returned onto a road, the position is uncertain. Therefore, the particles are spread in a wide area on the road, making the filter perform a soft restart.
5.3 Finding the Road Again

To evaluate the off-road handling a number of test-cases, including both off-road scenarios and scenarios without off-road driving, were run in a Monte Carlo simulation with 100 simulations per run. The results can be seen in Table 5.1. The test runs are shown in Figures C.2, C.3 and C.4 in Appendix C. In the first test run, the “back on track” field is not applicable, since the test drive was intended to test the detection mechanism, so it ended off-road. The second test run included an off-road drive of about 300 metres, where a lot of turns were made, making the test drive relatively difficult for the algorithm. The third test drive was a long drive with a few map incorrectnesses, intended to test the off-road mechanism for false alarms. The “back on track” field shows how many of the false alarms that was handled correctly. This means that two of the simulations did not solve the positioning problem as they should. Without off-road handling all runs were run without any problems. This shows that the thresholds might have to be raised a bit to decrease the number of false off-road alarms.

Table 5.1. Monte Carlo evaluation of off-road handling. Each test-run was simulated 100 times.

<table>
<thead>
<tr>
<th></th>
<th>#false alarms</th>
<th>#off-road detections</th>
<th>#back on track</th>
</tr>
</thead>
<tbody>
<tr>
<td>run1 (off road)</td>
<td>0</td>
<td>94</td>
<td>-</td>
</tr>
<tr>
<td>run2 (off road)</td>
<td>0</td>
<td>100</td>
<td>70</td>
</tr>
<tr>
<td>run3 (false alarm)</td>
<td>10</td>
<td>-</td>
<td>8</td>
</tr>
</tbody>
</table>

Figure 5.3. Off-road algorithm, main principles
As with the tuning of filter parameters, it is very important to fine tune the thresholds for off-road detection to get good performance. Another important factor is the choice of resspreading technique. It is easy to tune the parameters to suite a particular test run, but the trick is to find settings that is tolerant to map deficiencies, but also sensitive enough to detect off-road driving in all possible scenarios.

Figure 5.4. The vehicle has turned right onto a non existing road after about 600 samples.
Figure 5.5. The road ends dead ahead after about 160 samples.
This chapter will deal with the quantization problem. Quantization can be compared to sampling, but in the amplitude domain. When implementing the quantization a useful method is to use fixed point arithmetics which also will be explained in this chapter. The MAP algorithm is thought to be run on a hand-held computer without a floating point unit. Therefore the most time consuming floating point routines have been implemented using fixed point arithmetics and the results are presented here.

6.1 Quantization

Quantization is the discretization process of mapping a signal sequence of continuous amplitude onto a sequence with discrete amplitude values. This is very common since computers store everything as discrete values. Suppose that the value range $[-M/2, M/2]$ is divided into $N$ uniformly spread values, see Figure 6.1. This

![Figure 6.1. Input output characteristics of a uniform quantizer with step size $q$.](image-url)
Optimization

gives a quantization step, \( q = \frac{M}{N} \). For an arbitrary value, within the range, that is mapped onto the nearest quantization value, a quantization error of at most \( q^2 = \frac{M^2}{N^2} \) is made. In [20] the quantization error is modeled as a uniformly distributed, \( U[-\frac{q}{2}, \frac{q}{2}] \), random noise. This is correct under certain circumstances and approximately right otherwise. See [20] for a more detailed explanation. The effect is that in order to calculate the moments of the non-quantized signal given the quantized signal, the result has to be corrected using Sheppard’s corrections [20]

\[
\begin{align*}
\mathbb{E}\{x\} &= \mathbb{E}\{x_{\text{quant}}\} - 0 \\
\mathbb{E}\{x^2\} &= \mathbb{E}\{x^2_{\text{quant}}\} - \frac{q^2}{12} \\
\mathbb{E}\{x^3\} &= \mathbb{E}\{x^3_{\text{quant}}\} - \frac{2^2 \mathbb{E}\{x_{\text{quant}}\}}{4}.
\end{align*}
\]

For instance, consider a variable in the range \([0, 2000]\) whose value is quantized with 16 bits, that is \(2^{16} = 65536\) values. The quantization step is \(\frac{2000}{2^{16}} \approx 0.03\). To get the correct expected value of the variance, (assuming zero mean), the following adjustment have to be made

\[
\text{Var}(x) = \mathbb{E}\{x^2\} = \mathbb{E}\{x^2_{\text{quant}}\} - \frac{0.03^2}{12}.\]

The error in the variance calculation is relatively small, given that the quantization steps are small. When quantizing the particle filter, all of the quantization steps have been chosen small enough to be able to neglect this effect.

6.2 Fixed Point Representation

A real number is often referred to as a floating point number because of the fact that the decimal point can be placed arbitrarily, it “floats”, hence the name. Regular floating point numbers have an exponent, that describes the magnitude, and a mantissa that contains the actual value.

The problem with this approach is that algebraic operations becomes more complex than the same operations for integer based numbers. For instance, adding the floating point numbers 23.984 and 0.01 is a lot more demanding than adding the integers 23984 and 10 simply because there is a need to keep track of where the decimal point is.

One way to get around this is to use fixed point arithmetics instead. Fixed point arithmetics uses integers to represent real numbers. These integers are just scaled versions of the real number. When using the fixed point numbers it is just to use them as if they were real numbers, keeping the scale factor in mind. The scaling of the variables can be seen as predefining where the decimal (or binal) point is placed. How to chose this position is best explained with an example.
6.2 Fixed Point Representation

Example

Consider the following calculation

\[ \Delta s = T_s \left( R_n (w_{rr} + w_{rl}) - \frac{\delta (w_{rr} - w_{rl})}{2} \right). \] (6.3)

All the involved variables are floating point numbers. To choose where to place the binary point, in other words how much to scale, the ranges and precisions of the involved variables need to be known

\[ R_n = 0.29 \text{ (constant)} \]
\[ T_s = 0.5 \text{ (constant)} \]
\[ w_{rr}, w_{rl} \in [0, 500], \text{ resolution } \approx 10^{-2} \] (6.4)
\[ \delta \in [-0.05, 0.05], \text{ resolution } \approx 10^{-6} \]
\[ \Delta s \in [0, 150], \text{ resolution } \approx 10^{-2}. \]

The wheel speeds \( w_{rr} \) and \( w_{rl} \) need a resolution of about \( 10^{-2} > 2^{-7} \) meaning that at least seven bits are required to store the fractional part of the variables. To store the integral part nine bits are required, since \( 500 < 2^9 \). This implies that a total of 16 bits would be sufficient to represent these variables.

The tire radius difference \( \delta \) has a resolution of \( 10^{-6} > 2^{-20} \) which would imply 20 bits and one bit is needed for the sign. However, since the maximum value of \( \delta \) is \( 0.05 \lesssim 2^{-4} \), three bits can be saved, resulting in a total of 18 bits. The architecture on a normal computer system supports word-lengths of 8, 16, 32 or 64 bits and this means that the \( \delta \) variable gets 32 bits since 16 is not enough. Here there is a tradeoff to make, either lowering the demands on the accuracy or using more bits than really required. There is another advantage with using a shorter word-length, since the product of two numbers of \( N \) bit factors gives a \( 2N \) bit result. Some processors can not handle 64 bit word length so this can be a good reason to lower the accuracy demands. Of course this depends on the application. Choosing a 16 bit word length gives an accuracy of \( 2^{-15-3} \approx 4 \cdot 10^{-6} \). The same reasoning leads to \( \Delta s \) demanding 15 bits.

The constant 0.29 will in this example be scaled with a factor \( 2^{16} \), that is, 16 bit representation will be used. The reason for this is that an 8 bit representation would lead to a quantization error of \( \Delta s \approx 10^{-3}m \) that is a little too much. The other constant can be seen as a division by 2, which is easy to implement as a right shift.
Now that the fixed point formats are set as
\[
R_{n,\text{int}} = \left\lfloor (0.29 \cdot 2^{16}) \right\rfloor = 19005
\]
\[
T_{s,\text{int}} = 2^{-1} \text{(single right shift)}
\]
\[
w_{rr,\text{int}}, w_{rl,\text{int}} = \underbrace{xxxxxxxxxxxx} \quad (6.5)
\]
\[
\delta_{\text{int}} = 0.000\underbrace{xxxxxxxxxxxxxx}
\]
\[
\Delta s_{\text{int}} = \underbrace{xxxxxxxxxxxxxxx}
\]
\[x\] represents a value bit
\[s\] represents a sign bit,
on to the implementation part. To implement (6.3) using fixed point arithmetics, scale the variable according to their fixed point format. Then simply calculate the equation as it is, scaling the part results so that they agree with the fixed point format of the result. For instance, if \(w_{rr} = 23.5467\) the fixed point version will be \(w_{rr,\text{int}} = \left\lfloor 23.5467 \cdot 2^7 \right\rfloor = 3013\) and so on.

\[
\Delta_{s,\text{int}} = T_{s,\text{int}}(R_{n,\text{int}}(w_{rr,\text{int}} + w_{rl,\text{int}}) - \delta_{\text{int}}(w_{rr,\text{int}} - w_{rl,\text{int}}) \cdot 2^{-1})
\]
\[
= 19005(w_{rr,\text{int}} + w_{rl,\text{int}}) \cdot 2^{-(16+1+7+1-8)}
\]
\[
- \delta_{\text{int}}(w_{rr,\text{int}} - w_{rl,\text{int}}) \cdot 2^{-(18+7+1+1-8)}
\]
\[
= 19005(w_{rr,\text{int}} + w_{rl,\text{int}}) \cdot 2^{-17} - \delta_{\text{int}}(w_{rr,\text{int}} - w_{rl,\text{int}}) \cdot 2^{-19}
\]

(6.6)

For some values on \(\delta, w_{rr}\) and \(w_{rl}\) the fixed point implemented version is compared to the floating point version. The result is presented in Table 6.1 and they show that the accuracy is as high as expected, that is an error approximately less than \(2^{-8} \sim 0.004\).

<table>
<thead>
<tr>
<th>(w_{rr}), (w_{rl}), (\delta)</th>
<th>Floating point (\Delta s)</th>
<th>Fixed point (\Delta s_{\text{int}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.23, 20.57, 0.001</td>
<td>2.958</td>
<td>(\sim 2.957 \cdot 2^{-8})</td>
</tr>
<tr>
<td>90.1, 100.4, 0.01</td>
<td>13.837</td>
<td>(\sim 13.832 \cdot 2^{-8})</td>
</tr>
<tr>
<td>3.29, 4.60, 0.03</td>
<td>0.575</td>
<td>(\sim 0.578 \cdot 2^{-8})</td>
</tr>
</tbody>
</table>

### 6.3 Bottlenecks in the Algorithm

Before optimizing the algorithm, for instance by using fixed point arithmetics, the bottlenecks of the system need to be located, since this is where the largest improvements of calculational burden can be made. This was done by timing different parts in the algorithm and a “profile” of the algorithm was built. The profile was done on the target platform, a Compaq iPAQ (see Figure 6.2), that lacks a floating point unit. As can be seen in Figure 6.3 the main bottlenecks lie in the calculation of the shortest distance to the closest road, the updating of the weights and the time update of the particles. All these function contain quite a few floating point operations.
6.4 Implementation

All the details in the actual implementation will not be covered, but some remarks about some of the choices during the process can be made. The particle weights are the most important factor in the particle filter. They also tend to become very small, especially with many particles, so a high resolution is needed for the weights. In this case, a 32 bit resolution was chosen to represent the weights giving a resolution of at least $3 \cdot 10^{-10}$.

The vehicles direction, $\psi$, is bounded between 0 and $2\pi$ rad. The choice here is whether to have a resolution of 8 or 16 bits which corresponds to an accuracy of $1.4$ or $0.005$ degrees. A resolution of $1.4$ degrees gave an unstable filter so the latter was chosen. Now that only a discrete number of directions can be realized, it is a good idea to make a table of all the corresponding cosine and sine values for these, speeding up these calculations considerably.

6.4.1 Results from the Quantized Particle Filter

The particle filter simulates a probability density by updating importance weights. These weights are all in the range $[0, 1]$, which makes this problem suitable for fixed point implementation. Other bottlenecks are the time update and the calculation of the shortest distance to the closest road. By making all particle states integer based and also making integer based versions of some non-linear functions, such as the arcus tangent and the exponential function, a major decrease in the computational burden was obtained. The results are shown in Figure 6.3. The fixed point version of the code can cope with ten times more particles compared to the floating point implementation, giving a tenfold performance increase. Fixed point implementation implies quantization which means a little less accuracy. However, the gain in number of particles is a lot more important than the small loss in accuracy due to quantization effects.
Figure 6.3. The average completion times for some of the most time critical functions.

MDC : The calculation of the minimum distance to the closest road.
Residues : The calculation of the residuals used to update the weights.
Weights : The updating of the weights.
TU : The time update of the particle filter.
Graphics : The time for the Graphical Users Interface (GUI) to complete its tasks.
Total : The average time for one filter iteration.
Chapter 7

Conclusions and Future work

7.1 Conclusions

This thesis has been focusing on further developing map-aided positioning using some new filter approaches and by handling new scenarios, such as off-road driving.

The auxiliary particle filter (APF) has been implemented and the evaluations show no significant improvement. The reason for this is that the APF improves the performance mainly on systems where the measurements are very informative, since the idea is to look one measurement ahead. In the map-aided positioning case, the most information is received when the vehicle turns. The measurements are in this case map database information that is digitized so that the turns are sharp and edgy, making the measurements uncertain.

By applying Rao-Blackwellization techniques and separating the linear parts of the state space model from the non-linear, a filter with a slightly better robustness was obtained. This filter was more computational expensive and the result was therefore that there was no gain in the tradeoff between robustness and computational load.

The reason for filter divergence is in most cases map deficiencies or lack of map information. Both these cases can be handled as off-road scenarios. Therefore, an algorithm that can handle off-road driving will be more robust. In this thesis an attempt has been made to solve this problem. A mechanism for detecting off-road driving has been developed. The results show that this mechanism performs really well. A technique of finding the way back onto the road again has also been implemented and tested. The conclusions that can be drawn is that it is possible to have support for shorter off-road drives in the algorithm.

The algorithm has been successfully implemented on a small hand-held computer together with a simple navigator. This proves that it is possible to build a complete navigation system using relatively simple and cheap equipment.
7.2 Future Work

Future work on this product is to create a solution to the problem of finding an initial search area. As a stand-alone positioning system, the user is today required to provide the initial guess. The third generation of cellular phones, 3G, are beginning to hit the market and their rough positioning should be adequate as an initial search area for MAP. Another possibility could be to triangulate RDS-radio signals to get a position estimate.

More work needs to be performed on the off-road handling. The thresholds for off-road detection has to be fine tuned and to support longer off-road drives, the method of restarting once on-road again has to be improved. Another important issue that needs to be resolved is how to choose an optimal starting estimate for the off-road drive using the dead reckoning estimates.

In the initial phase, when the particles are spread out over a large area of the map, there is a high degree of clustering. More work should be put into finding algorithms for dealing with clusters that are separated in space. Each cluster could perhaps be handled with a separate filter.
Bibliography


Conclusions and Future work


Appendix A

Spring Tire Model

Consider a vehicle that is turning with a constant yaw rate $\dot{\psi}$ and turning radius $R$. The relation between the (constant) speed of the vehicle $V$, $R$ and $\dot{\psi}$ can in the body fixed frame be expressed as

$$V(2) = R\dot{\psi}\hat{x}_2.$$  \hspace{1cm} (A.1)

Figure A.1. The reference frames used in the derivation.

With two reference frames defined as in Figure A.1 the acceleration of the vehicle
can be calculated as a cross product, using the Coriolis acceleration equations [9]

\[
a_{(2)} = \frac{d}{dt}(V_{(2)})_{(1)} + \Omega \times V_{(2)} = \begin{pmatrix} 0 \\ 0 \\ -\dot{\psi}R \end{pmatrix} + \begin{pmatrix} -\dot{\psi} \dot{R} \\ 0 \\ 0 \end{pmatrix}_{(2)} = \begin{pmatrix} 0 \\ -\dot{\psi}^2 R \\ 0 \end{pmatrix}_{(2)}. \quad (A.2)
\]

This means that, as expected, a lateral acceleration will affect the vehicle when turning. The acceleration can be seen as a force \( F_a = ma \) acting upon the vehicle. In equilibrium the resulting force on the vehicle should be zero and this gives the equation

\[
\uparrow : N1 + N2 = mg \quad \text{(A.3a)}
\]

\[
\bowtie : \frac{1}{2} mah_{CoG} = (N1 - N2)L, \quad \text{(A.3b)}
\]

where \( N1, N2 \) are the normal forces from the tires, \( m \) is the mass of the vehicle, \( h_{CoG} \) is the height to the center of gravity and \( L \) is the width of the rear wheel axis. Interesting for the tire radius difference is \( N1 - N2 \) which is directly given by (A.3b) as

\[
(N1 - N2) = \frac{2mah_{CoG}}{L} = \left\{ a = -\dot{\psi}^2 R = -\dot{\psi}V \right\} = -\frac{2m\dot{\psi}Vh_{CoG}}{L} \quad \text{(A.4)}
\]

Assuming that each tire can be modeled as an ideal spring, with spring constant \( k_{tire} \), the tire radius difference due to turning can be expressed as

\[
\delta r' = -\frac{2m\dot{\psi}Vh_{CoG}}{k_{tire}L}. \quad \text{(A.5)}
\]
Figure B.1. The computed acceleration signal compared to the $\delta^*_{rl}$ estimate.
Figure B.2. The acceleration compensated signal is a little bit more constant than the original.
Appendix C

Testruns

Figure C.1. Testrun 1. This is a long testdrive used for robustness checks.

Figure C.2. Testrun 2. This testrun includes an off-road drive.
Figure C.3. Testrun 3. This testrun includes an off-road drive on a parking lot. The vehicle drives back onto the road after the off-road drive.

Figure C.4. Testrun 4. This is a long testdrive used for robustness and false alarm checks.