Linköping studies in science and technology. Dissertations.
No. 1216

Estimation and Detection with Applications to Navigation

David Törnqvist

Department of Electrical Engineering
Linköping University, SE–581 83 Linköping, Sweden
Linköping 2008

Linköping studies in science and technology. Dissertations.
No. 1216

Estimation and Detection with Applications to Navigation

David Törnqvist

`tornqvist@isy.liu.se`
`www.control.isy.liu.se`
Division of Automatic Control
Department of Electrical Engineering
Linköping University
SE–581 83 Linköping
Sweden

ISBN 978-91-7393-785-6 ISSN 0345-7524

Copyright © 2008 David Törnqvist

Printed by LiU-Tryck, Linköping, Sweden 2008

To my family!

Abstract

The ability to navigate in an unknown environment is an enabler for truly autonomous systems. Such a system must be aware of its relative position to the surroundings using sensor measurements. It is instrumental that these measurements are monitored for disturbances and faults. Having correct measurements, the challenging problem for a robot is to estimate its own position and simultaneously build a map of the environment. This problem is referred to as the Simultaneous Localization and Mapping (SLAM) problem. This thesis studies several topics related to SLAM, on-board sensor processing, exploration and disturbance detection.

The particle filter (PF) solution to the SLAM problem is commonly referred to as FastSLAM and has been used extensively for ground robot applications. Having more complex vehicle models using for example flying robots extends the state dimension of the vehicle model and makes the existing solution computationally infeasible. The factorization of the problem made in this thesis allows for a computationally tractable solution.

Disturbance detection for magnetometers and detection of spurious features in image sensors must be done before these sensor measurements can be used for estimation. Disturbance detection based on comparing a batch of data with a model of the system using the generalized likelihood ratio test is considered. There are two approaches to this problem. One is based on the traditional parity space method, where the influence of the initial state is removed by projection, and the other on combining prior information with data in the batch. An efficient parameterization of incipient faults is given which is shown to improve the results considerably.

Another common situation in robotics is to have different sampling rates of the sensors. More complex sensors such as cameras often have slower update rate than accelerometers and gyroscopes. An algorithm for this situation is derived for a class of models with linear Gaussian dynamic model and sensors with different sampling rates, one slow with a nonlinear and/or non-Gaussian measurement relation and one fast with a linear Gaussian measurement relation. For this case, the Kalman filter is used to process the information from the fast sensor and the information from the slow sensor is processed using the PF. The problem formulation covers the important special case of fast dynamics and one slow sensor, which appears in many navigation and tracking problems.

Vision based target tracking is another important estimation problem in robotics. Distributed exploration with multi-aircraft flight experiments has demonstrated localization of a stationary target with estimate covariance on the order of meters. Grid-based estimation as well as the PF have been examined.

Populärvetenskaplig sammanfattning

Tänk dig att du är en brandman på väg in i ett eldhärjat hus. Du vet inte hur det ser ut där inne eller vilka vägar som är möjliga att ta. Väl inne i huset utforskar du det på jakt efter brandorsaken. Det är samtidigt mycket viktigt att komma ihåg vilken väg du kom och hur långt det är till närmaste utgång om du snabbt behöver lämna huset. Under utforskandet byggs en mental karta över huset upp genom att använda synintryck och balanssinne. Ibland kanske du återkommer till en bekant plats och kan då uppdatera din karta lite extra eftersom du förstår hur olika delar av huset hänger ihop.

Eftersom vi vill undvika att människor skadas i samband med att de ska släcka bränder eller utföra andra farliga uppdrag kommer det i framtiden bli vanligt med robotar som hjälper oss med dessa uppdrag. Scenariot ovan är därför något som framtida robotar kommer ställas inför. De här robotarna måste förstås ha sensorer för att känna av omgivningen och sin egen rörelse. Sensorer som används för detta är bland annat kameror, laser, radar, mikrofon, kompass, accelerometer och gyroskop. Från dessa sensorer flödar mängder med information som måste utnyttjas på ett smart sätt för att roboten ska bli så framgångsrik som möjligt. Det är denna hantering av information som den här avhandlingen behandlar.

I det inledande scenariot handlade det mycket om att försöka bygga upp en karta över huset. Inom robotiken kallas detta problem för *samtidig lokalisering och kartuppbyggnad* och förkortas SLAM (eng. Simultaneous Localization And Mapping). SLAM-problemet har studerats flitigt under de senaste tjugo åren, men det kräver mycket beräkningar vilket gör att datorer inte klarar av att bygga särskilt stora kartor. Mycket forskning görs därför på hur man kan förenkla problemet så att mindre beräkningar krävs. Ett bidrag i avhandlingen behandlar ett sätt att minska beräkningsbördan för robotar som kräver avancerade modeller. I det här fallet används en helikopter som är utrustad med en kamera, accelerometer, gyroskop och barometer. Helikoptern flyger över ett område och tittar ner på marken med kameran. Intressanta objekt i bilden sparas och bygger upp kartan som helikoptern kan använda för att positionera sig med. Processen att bygga upp kartan och positionera sig bygger på så kallade *estimeringsmetoder*.

För att välja ut särskilt intressanta objekt i bilden, som lämpar sig för att bygga en karta med, används olika bildbehandlingsmetoder. De försöker välja ut objekt som ska vara lätta att känna igen från en bild till en annan. Ibland blir det fel då metoderna luras av ett snarlikt objekt i bilden. För att upptäcka den här typen av fel används *statistiska detektionsmetoder*. Ett annat område där dessa metoder kan användas är för att upptäcka fel i sensormätningar. Ibland används en kompass för att veta i vilken riktning roboten färdas. Den är mycket känslig för störande magnetfält och dessa störningar måste därför upptäckas.

Ett annat område som behandlas i avhandlingen är sökning efter t.ex. en försvunnen person med hjälp av flera obemannade flygplan. Flygplanen är utrustade med kameror som spanar ner på marken. Estimeringsmetoder används för att representera information om vad flygplanen har sett och inte har sett så att den kan distribueras mellan flygplanen. Tanken med tekniken är att flygplanen ska få beslutsunderlag för att själva kunna fatta beslut om var det är lämpligt att söka.

Sammanfattningsvis kan man säga att den här avhandlingen syftar till att robotar ska kunna använda sina “ögon”, “öron” och andra sensorer genom utnyttjande av estimerings- och detektionsmetoder.

Acknowledgments

When crossing a river, you must make sure to find good rocks to stand on. My rocks when writing this thesis has been Prof Fredrik Gustafsson and lately Dr Thomas Schön. Fredrik has a constant flow of ideas and can find a seed to something positive in everything. Thomas, discussing with you is great and you have provided invaluable feedback during this work. Thank you both!

I would also like to thank Prof Lennart Ljung for drafting me to the control group and for creating a good atmosphere in the group. Our secretary Ulla Salaneck also deserves my gratitude for always arranging practical matters in a smooth way.

Various parts of this thesis have been proofread by Fredrik, Dr Gustaf Hendeby, Dr Umut Orguner, Thomas, Per Skoglar, Martin Skoglund, and Lic Henrik Tidefelt. Your comments have been invaluable! Thank you! Gustaf also deserves appreciation for answering all these LaTeX-questions! Henrik helped me with some of the figures, thanks!

During 2007, I had the opportunity to visit two different research centers for a couple of months each. First, I visited the Australian Center for Field Robotics. This was really inspiring, seeing all the different airplanes, helicopters and ground robots. The people at the department were great, they learned me how to appreciate good pub food (especially those burgers with beetroot), they made great espresso and we had nice discussions about SLAM. Thank you! After this visit, I was invited to C3UV at University of California, Berkeley. That was a great time flying helicopters and airplanes, experiencing the desert heat and discussing particle filters. I really liked to work with you there, thank you!

The four of us who started our PhD studies at the same time, we did this journey together. Thank you Dr Daniel Axehill, Gustaf and Dr Johan Sjöberg for company during many late nights at the office, spare time activities, and for simply being really good friends! I would also like to thank all the other people at the Automatic Control group for the good atmosphere and for all the laughs created by fika-discussions and junk-mail.

My co-authors in different papers over the years are Dr Gunnar Bark, Lic Gianpaolo Conte, Dr Eva Englund, Dr Erik Geijer Lundin, Dr Fredrik Gunnarsson, Prof Fredrik Gustafsson, Prof Karl Hedrick, Dr Rickard Karlsson, Dr Zu Kim, Ass Prof Inger Klein, Allison Ryan, Dr Thomas Schön, John Tisdale, and Dr Niclas Wiberg. Thank you very much for inspiring collaborations!

This work was partially sponsored by SSF strategic research center Modeling, Visualization and Information Integration MOVIII, the Center for Industrial Information Technology, CENIIT, and the Excellence Center in Computer Science and Systems Engineering in Linköping, ECSEL, which are hereby gratefully acknowledged.

Finally, I would like to thank my family for always believing in me. It is really great to feel the support you are giving me!

Linköping, October 2008
David Törnqvist

Contents

1 Introduction **1**
1.1 Robots 1
1.2 Robotic Structure 3
    1.2.1 Sensors 4
    1.2.2 Sensor Fusion 5
    1.2.3 Monitoring 6
    1.2.4 Planning and Control 7
1.3 Outline 7
    1.3.1 Outline of Part I 7
    1.3.2 Outline of Part II 8

I Background Theory **13**

2 Modeling **15**
2.1 System Descriptions 15
    2.1.1 State Space Model 15
    2.1.2 Batched Systems 16
    2.1.3 Structured Faults 17
2.2 Rigid Body Modeling 18
    2.2.1 Coordinate Systems 19
    2.2.2 Dynamic Model 19
    2.2.3 Measurements 21
    2.2.4 Discrete-Time Dynamic Model 22
    2.2.5 Linearization 24

3 Estimation and Detection Theory **25**
3.1 Bayesians and Frequentists 25
3.2 Parameter Estimation 26
3.3 Recursive Bayesian Filtering 27
3.4 Kalman Filter 29
3.5 Extended Kalman Filter 30
3.6 Particle Filter 31
3.6.1 Derivations 32
3.6.2 Discussion 34
3.7 Rao-Blackwellized Particle Filter 36
3.7.1 Derivations 36
3.8 Hypothesis Testing 39
3.8.1 Simple Hypothesis Testing 41
3.8.2 Composite Hypothesis Testing 42

4 Simultaneous Localization and Mapping **45**
4.1 Problem Formulation 45
4.2 EKF-SLAM 48
4.2.1 Algorithm 49
4.2.2 Simulation Example 51
4.3 FastSLAM 52

5 Concluding Remarks **55**
5.1 Results 55
5.2 Future Work 56

A Probability Distributions **59**
A.1 Gaussian Distribution 59
A.2 Chi-square Distribution 60
A.2.1 Central Chi-square 60
A.2.2 Noncentral Chi-square 61

B Quaternion Preliminaries **63**
B.1 Prerequisites in Vector Kinematics and Mathematics 63
B.1.1 Cross-product 63
B.1.2 Vector Rotation 64
B.1.3 Direction Cosines 64
B.2 Operations and Properties of Quaternions 65
B.3 Describing a Rotation with Quaternions 66
B.4 Rotation Matrix 67
B.5 Dynamics of Quaternions 68

C Code for SLAM-Example **71**

Bibliography **73**

II Publications 79

A Particle Filter SLAM with High Dimensional Vehicle Model 81
- 1 Introduction . . . 83
- 2 Problem Formulation . . . 85
- 3 Particle Filter for SLAM Utilizing Structure . . . 86
  - 3.1 Algorithm . . . 87
  - 3.2 Likelihood Computation . . . 89
  - 3.3 Map Estimation and Map Management . . . 90
- 4 Application Example . . . 91
  - 4.1 Model . . . 91
  - 4.2 UAV Platform . . . 95
  - 4.3 Experiment Setup . . . 95
  - 4.4 Experimental Results . . . 96
- 5 Conclusion . . . 98
- A Appendix . . . 98
  - A.1 Coordinate Systems . . . 98
- References . . . 99

B Unifying the Parity-Space and GLR Approach to Fault Detection with an IMU Application 103
- 1 Introduction . . . 105
- 2 Modeling . . . 107
  - 2.1 System Model . . . 107
  - 2.2 Batch Model . . . 107
  - 2.3 Fault Model . . . 108
  - 2.4 Model Summary . . . 108
- 3 State Estimation . . . 109
  - 3.1 Least Squares Estimation . . . 109
  - 3.2 Minimum Variance State Estimation . . . 110
  - 3.3 Filtered Estimate from Past Data . . . 110
  - 3.4 Smoothed State Estimate . . . 110
- 4 Fault Detection with Parity Space . . . 111
  - 4.1 Classical Parity Space . . . 111
  - 4.2 Parity Space as Least Squares State Estimation . . . 111
  - 4.3 Parity Space with Smoothing . . . 112
- 5 GLR Test . . . 113
  - 5.1 Test Statistic . . . 113
  - 5.2 Statistics . . . 113
- 6 Simulation Example . . . 114
  - 6.1 Modeling . . . 114
  - 6.2 Simulations . . . 114
  - 6.3 Fault Detection . . . 115
- 7 Application Example: Detection of Magnetic Disturbances . . . 116
  - 7.1 Modeling . . . 118
  - 7.2 Detection Algorithm . . . 120

7.3 Test Results . . . 120
8 Conclusions and Discussion . . . 125
A Appendix . . . 125
A.1 Proof of Lemma 1 . . . 125
References . . . 125

C Detecting Spurious Features using Parity Space 127
1 Introduction . . . 129
2 Parity Space . . . 130
2.1 Modeling . . . 131
2.2 Residual Generation . . . 132
2.3 Statistical tests . . . 132
3 Detection of Spurious Features . . . 133
3.1 Known Translation . . . 134
3.2 Unknown Translation . . . 135
4 Simulations . . . 136
4.1 Known Translation . . . 137
4.2 Unknown Translation . . . 138
5 Real World Experiment . . . 138
6 Conclusions and Future Work . . . 139
References . . . 139

D Fast Particle Filters for Multi-Rate Sensors 141
1 Introduction . . . 143
2 Nonlinear State Filtering . . . 145
2.1 Conceptual Solution . . . 145
2.2 Particle Filter . . . 146
3 Multi-rate Particle Filter . . . 147
3.1 Algorithm . . . 147
3.2 Computing Estimates . . . 150
4 Simulations . . . 150
4.1 Setup . . . 151
4.2 Simulation Results . . . 151
5 Conclusion . . . 154
References . . . 154

E A multiple UAV system for vision-based search and localization 157
1 Introduction . . . 159
2 Decentralized Filtering for Search and Track . . . 160
3 Modeling and Implementation . . . 161
3.1 Target Localization Model . . . 162
3.2 Target Detection Model . . . 163
3.3 Target Model . . . 164
4 Experimental Results and Analysis . . . 164
4.1 Grid-based Estimation . . . 165
4.2 Particle Filter Implementation . . . 166

    4.3 Error Analysis and Comparison 168
5 Conclusions 168
References 170

1

Introduction

IMAGINE A WORLD where our cars drive without a driver and the airplanes do not have pilots. In such a world, you can relax with a movie or read an interesting book while the car drives you where you want to go. In such a world, lost people in a catastrophe area can be found thanks to hundreds of small and inexpensive unmanned airplanes that easily can search the area. The applications above are examples of autonomous systems. Systems with some degree of autonomy are already used today, for example robots already build products in our factories, clean our floors and drive our trains. An important enabler for autonomous systems is the ability to navigate in an unknown or partially known environment and to do this reliably without support of external infrastructure. Estimation and detection methods are essential building blocks for such a navigation system, and the topics of this thesis.

This introductory chapter is divided into two parts. The first part gives some robotics background and puts the publications included in the thesis in a bigger context. The second part discusses the outline of the thesis and the author's contributions.

1.1 Robots

The word *robot* is Czech for work or labor and was first mentioned in todays context in a play by Karel Čapek, written in the 1920s. In this play human-like mechanical creatures, robots, were created to work in the factories. Today, a robot is a machine that has some level of autonomy. Some of them are autonomous and make their own decisions while others are preprogrammed to perform a certain task. A few examples of robots will be given in this section.

An industrial robot (shown in Figure 1.1(a)) consists of a mechanical arm with a number of joints, capable of doing welding, painting, packaging and other parts of manufacturing traditionally carried out by humans. This has revolutionized the manufacturing process. Monotonous and dangerous work can now be done fast, at low cost, and with

***(a)** Industrial robot. By courtesy of ABB.*

***(b)** Autonomous vacuum cleaner. By courtesy of Electrolux.*

Copyright Saab AB,
Photo: peter@svarteld.com

***(c)** Autonomous car. By courtesy of Tartan Racing.*

***(d)** Autonomous helicopter. By courtesy of Saab.*

***Figure 1.1:** Examples of robots.*

high precision.

The task of many robots is to make life easier for humans. One such robot is the autonomous vacuum cleaner, shown in Figure 1.1(b). This robot will explore the room and then move in a strategic pattern to clean the room as fast as possible. When batteries get low, it automatically returns to the home base to recharge.

One of the success stories in creating autonomous systems is in the DARPA Grand Challenge. This is a competition for autonomous cars organized by Defense Advanced Research Projects Agency (DARPA), which is the military research agency in the US. While the first two competitions were located in the desert, the last competition in 2007 took place in an urban environment and is known as the *Urban Challenge*. The course of the competition was 97 km long and the participating autonomous cars had to obey all traffic rules. As many as 89 teams from all over the world entered the competition and six of them reached the goal. The winning car from team Tartan Racing at Carnegie Mellon University can be seen in Figure 1.1(c). The Journal of Field Robotics have had two special issues on the Urban Challenge in 2008, where the winning team presents their work in Urmson et al. (2008).

There are many active research topics concerning Unmanned Aerial Vehicles (UAVs). Today, UAVs are mainly used in the military for surveillance and reconnaissance. For the future, many civilian applications can be imagined. Searching for lost people, inspecting power lines, border patrolling, traffic surveillance and wildfire detection are only a few of the applications. The unmanned helicopter in Figure 1.1(d) is developed by the Swedish company Saab AB. It is autonomous and intended to be used for both military and civilian applications. One of the biggest challenges for the UAV community is to get UAVs certified for civilian airspace. In order to do this the UAV community will have to show, among other things, that the UAV always can navigate and find its way back home. Satellite navigation, such as GPS, Galileo and GLONASS, are excellent navigation systems. However, these kind of systems rely on an external infrastructure. That means there is a risk for outages due to, for example, satellite failure, jamming, solar storms or political reasons. Thus, there is a need for redundancy with alternative navigation systems. There are many techniques for terrain based navigation that can be used, some of them have a predefined map over the area and others build the map during flight. Human pilots for example, can recognize landmarks and navigate using a map. This sort of navigation system is needed for UAVs as well. Some aspects of terrain based navigation and map building is covered in Paper A.

1.2 Robotic Structure

Figure 1.2 provides an illustration of the high level structure of the information flows in a robot. To exemplify this structure, consider an autonomous helicopter. First of all this type of robot is unstable, which means that counteractions continuously have to be made in order to prevent the helicopter from drifting off track. Here is an obvious need for sensors, giving information about the helicopter's position and orientation. However, the individual sensors might not give the complete picture. They need to be combined using *sensor fusion* methods. When fusing sensor measurements, it is important to monitor the measurements for faults or disturbances. Using erroneous measurements could lead to fatal effects. Having information about the helicopter's position, orientation and perhaps also the surroundings, the desired path of the helicopter can be planned and the required control actions calculated. The control actions are then realized using the actuators, which in the case of a helicopter include the pitch angle of the blades and motor speed. The rest of this section will give more insight about the blocks introduced in Figure 1.2.

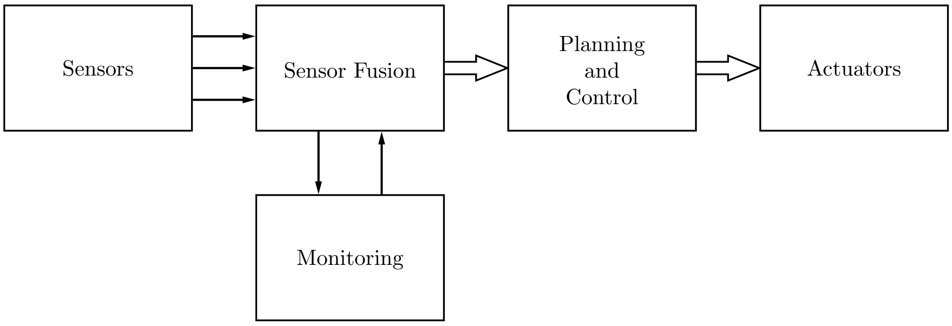
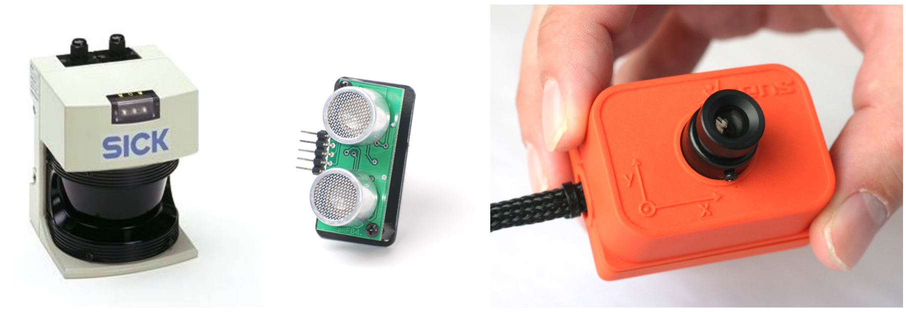


Figure 1.2: *Internal structure of an autonomous robot. The senors provide the sensor fusion algorithms with measurements. During the fusion process, the measurements are monitored to detect disturbances or faults. The outcome of the sensor fusion block is an estimate of the current state of the robot, and is passed on to the planning and control block. Finally, the actuators provide the physical control.*

1.2.1 Sensors

The autonomous car, discussed in Section 1.1, needs sensors to navigate and understand its surroundings. It uses a combination of sensors to get a broad coverage but also for redundancy. It has 16 laser based sensors, LIght Detection And Ranging (LIDAR) sensors, one such sensor is depicted in Figure 1.3(a). The LIDAR has a sweeping laser beam, giving information about heading and distance to an obstacle.



(a) *Laser scanner (LIDAR)* ***(b)*** *Sonar sensor* ***(c)*** *Combined IMU and camera, by courtesy of Xsens Technologies B.V.*

Figure 1.3: *Examples of sensors.*

The UAVs in Paper A, C and E use an Inertial Measurement Unit (IMU) and a camera. There are sensor units consisting of these sensors, one of them developed by Xsens Technologies B.V. and depicted in Figure 1.3(c). An IMU consists of accelerometers and gyroscopes, measuring accelerations and angular velocities.

The autonomous vacuum cleaner in Figure 1.3(b) uses sound navigation and ranging (sonar) to navigate. The technique of this sensor is to send out a sound pulse and then wait for the echo. The distance to an obstacle can then be calculated using the speed of sound. A typical sonar has two microphones making it possible to determine the angle of the echo as well. This type of sensor is cheap and simple to use and is therefore very popular for ground robot navigation. An example of a sonar sensor is shown in Figure 1.3(b).

Measurements from sensors that sense the environment, e.g., LIDAR, camera and sonar, have to be processed before the measurements can be used in a filter. For example in an image from a camera, *interest points* or *features* are found and then tracked in consecutive images. The interest points are found using detectors. One example of such a detector is the *Harris detector* (Harris and Stephens, 1988) which selects patches in an image with large contrast. An example of this, taken from Paper C, can be seen in Figure 1.4.

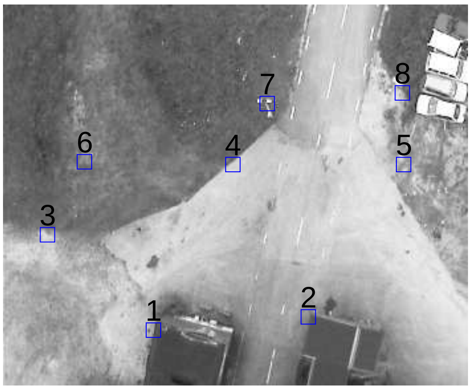


Figure 1.4: *Example of camera post-processing. The image shows the view from a helicopter at 60 m altitude. The numbered squares are patches selected using a Harris-detector.*

1.2.2 Sensor Fusion

The task of sensor fusion is to combine several sensor measurements or information sources to extract the best possible information about the state of a system, e.g., a robot. The state contains all information about the robot needed for predicting the future behavior of the robot. To be able to use the different sensors and also relate measurements in time, a model describing the relations is needed. The commonly used state-space model can be written as

$$x_{t+1} = f(x_t, u_t) + v_t, \tag{1.1a}$$

$$y_t = h(x_t, u_t) + e_t, \tag{1.1b}$$

where x_t denotes the state, u_t known control inputs, y_t the measurements, and v_t and e_t are stochastic process noise and measurement noise, respectively. Note here that (1.1a) describes how the state evolves over time (dynamics) and (1.1b) describes how the measurements are related to the current state of the system.

In estimation the model (1.1) and the available measurements are used to gain information about the state. The algorithms used to find this state are referred to as *estimators* or *filters* and are the main topic of Chapter 3. The estimators used in this thesis build on the well-known Kalman Filter (KF) and Particle Filter (PF). The KF (Kalman, 1960) applies when (1.1) is a linear system with Gaussian noises, giving a solution that is parameterized by state mean and covariance. Also nonlinear systems can be handled by making an approximation using linearization. This results in the popular Extended Kalman Filter (EKF). The PF (Gordon et al., 1993) is a non-parametric solution to the estimation problem, also capable of handling nonlinear systems. It uses samples of the state distribution, *particles*, which are updated according to the system model. The accuracy of this method depends on how many particles that are used. However, using many particles also increases the computational load.

One example of sensor fusion relating to robots is the use of IMU measurements for navigation. The dynamic model (1.1a) will explain that acceleration is the second derivative of position and angular velocity is the derivative of angles. The estimation problem will here be solved using a filter that basically integrates the measurements to get the position and orientation. However, integration of noisy measurements leads to drift in the estimates. This can be solved by adding more sensors, for instance a camera and a barometer as is done in Paper A.

Also in other work, several sensors are combined to get better precision. The obvious sensors in industrial robots are encoders measuring the joint angles. Knowing the robot configuration, the position of the tool can then be calculated. However, robots are somewhat flexible, giving rise to errors in the estimates. In the work by Karlsson and Norrlöf (2004), sensor fusion is used to combine the angular measurements with accelerometers, enhancing the information about the position.

1.2.3 Monitoring

A prerequisite for successful sensor fusion is that the measurements are correct. All sensors can fail and some sensors are very sensitive to disturbances. Monitoring the sensors to detect these changes is therefore instrumental. This can be done using a residual, which over a sliding window of length L is defined as

$$r_t = \begin{pmatrix} y_{t-L+1} \\ \vdots \\ y_t \end{pmatrix} - \begin{pmatrix} h(\hat{x}_{t-L+1|t-L}, u_{t-L+1}) \\ h(f(\hat{x}_{t-L+1|t-L}), u_{t-L+2}) \\ \vdots \\ h(f^{L-1}(\hat{x}_{t-L+1|t-L}), u_t) \end{pmatrix}, \tag{1.2}$$

where $\hat{x}_{t-L+1|t-L}$ is the prediction of the initial state x_{t-L+1} using measurements up to time $t - L$ and $f^{L-1}(\,\cdot\,)$ means that $f(\,\cdot\,)$ has been applied $L - 1$ times. The statistical properties of the residual can be analyzed and a statistical test can be performed to determine if a fault occurred or not. In the case of a linear system with Gaussian noise where the states are estimated using measurements from the window, the residual in (1.2) will be calculated using a projection of the measurements. This is the classical *parity space method* (Chow and Willsky, 1984).

One example of disturbance detection is given in Paper C, where spurious features are detected. The problem arises since it is sometimes hard to match features in two consecutive images. A feature that is erroneously matched is referred to as spurious. If the images are combined with more sensors and a model of the system, the system will become redundant. This basically means that we have information about the features from different sources. Using this information, redundancy relations can be formed which can be used for detection. The same framework is also used in Paper B to detect magnetic disturbances.

1.2.4 Planning and Control

An autonomous system must have the ability to plan ahead. For a UAV the planning could consist in calculating the optimal route for keeping track of two cars. Since the cars are constantly moving, the optimal route for tracking them changes as well. The cars might drive in different directions which makes it impossible to track both at the same time. When one car is tracked, the uncertainty about the other grows and vice versa. How to select which car to track will for example depend on the road network. If not tracked, the uncertainty will grow faster for a car driving on a road with many junctions than for a car on a road without junctions. How fast the UAV can fly and how sharp turns it can make also has to be considered. How to minimize the uncertainty about the positions of the cars is a very challenging planning and control problem.

Classical control algorithms that stabilize the UAV in the example above and make it follow a desirable trajectory are studied in the area of automatic control. These algorithms utilize knowledge about the dynamics of the system and act based on this knowledge and measurements. Classical planning has been done at a higher level without regard to dynamic or stochastic effects. A control algorithm has then been applied to follow the trajectory given by the planning. The areas of automatic control and planning are now merging, leading to treatment of both problems simultaneously. A textbook covering these problems is LaValle (2006).

1.3 Outline

This thesis consists of two parts. The first part gives background and overview of the subjects treated in this thesis. The second part contains the included publications.

1.3.1 Outline of Part I

Model definitions and descriptions that are used in this thesis are provided in Chapter 2. Measurement disturbances need to be detected to avoid errors in the estimation process.

Both estimation and detection is the subject of Chapter 3. The estimation methods are used to solve the Simultaneous Localization and Mapping (SLAM) problem in Chapter 4. Finally, Chapter 5 provides conclusions and possible directions for future work.

1.3.2 Outline of Part II

The edited papers, which are included in Part II, are presented in this section. Both summary and the contributions are given below.

Paper A: Particle Filter SLAM with High Dimensional Vehicle Model

> D. Törnqvist, T. B. Schön, R. Karlsson, and F. Gustafsson. Particle filter SLAM with high dimensional vehicle model. *Journal of Intelligent and Robotic Systems*, 2008b. Accepted for publication.

Summary: This work presents a particle filter (PF) method closely related to FastSLAM for solving the simultaneous localization and mapping (SLAM) problem. Using the standard FastSLAM algorithm, only low-dimensional vehicle models can be handled due to computational constraints. In this work an extra factorization of the problem is introduced that makes high-dimensional vehicle models computationally feasible. Results using experimental data from a UAV (helicopter) are presented. The algorithm fuses measurements from on-board inertial sensors (accelerometer and gyro), barometer, and vision in order to solve the SLAM problem.

Background and contribution: It was during discussions between the author and Thomas Schön about SLAM implementations for a helicopter application that the main idea of the paper was formulated. Initial implementations used data from a camera and an IMU mounted on an industrial robot and was presented in Schön et al. (2007a) and helicopter data was first used in Karlsson et al. (2008). The author contributed the majority of this work.

Paper B: Unifying the Parity-Space and GLR Approach to Fault Detection with an IMU Application

> D. Törnqvist and F. Gustafsson. Unifying the parity-space and GLR approach to fault detection with an IMU application. *Automatica*, 2008. Submitted.

Summary: Using the parity-space approach a residual if formed by applying a projection to a batch of observed data, and this is a well established approach to fault detection. Based on a stochastic state space model, the parity-space residual can be put into a stochastic framework where conventional hypothesis tests apply. In an on-line application, the batch of data corresponds to a sliding window and in this contribution we develop an improved on-line algorithm that extends the parity-space approach by taking prior information from previous observations into account. For detection of faults, the

Generalized Likelihood Ratio (GLR) test is used. This framework allows for including prior information about the initial state, yielding a test statistic with a significantly higher sensitivity to fault. Another key advantage with this approach is that it can be extended to nonlinear systems using an arbitrary nonlinear filter for state estimation, and a linearized model around a nominal state trajectory in the sliding window. We demonstrate the algorithm on data from an Inertial Measurement Unit (IMU), where small and incipient magnetic disturbances are detected using a nonlinear system model.

Background and contribution: Detection using parity space was the subject of the authors first task as a PhD student. The work resulted in a few theoretical papers (Törnqvist et al., 2005, Törnqvist and Gustafsson, 2006) and the detection of magnetic disturbances was first presented in the Licentiate Thesis (Törnqvist, 2006). The author contributed the majority of this work.

Paper C: Detecting Spurious Features using Parity Space

D. Törnqvist, T. B. Schön, and F. Gustafsson. Detecting spurious features using parity space. In *Proceedings of 10th International Conference on Control, Automation, Robotics and Vision*, Hanoi, Vietnam, Dec. 2008a. Accepted for publication.

Summary: Detection of spurious features is instrumental in many computer vision applications. The standard approach is feature based, where extracted features are matched between the image frames. This approach requires only vision, but is computer intensive and not yet suitable for real-time applications. We propose an alternative based on algorithms from the statistical fault detection literature. It is based on image data and an inertial measurement unit (IMU). The principle of analytical redundancy is applied to batches of measurements from a sliding time window. The resulting algorithm is fast and scalable, and requires only feature positions as inputs from the computer vision system. It is also pointed out that the algorithm can be extended to also detect non-stationary features (moving targets for instance). The algorithm is applied to real data from an unmanned aerial vehicle in a navigation application.

Background and contribution: When working with visual SLAM applications, the need for detecting spurious features is obvious. To do this in a structured way, methods from the statistical detection domain were applied. The paper is based on the methods derived in Törnqvist et al. (2005), Törnqvist and Gustafsson (2006) and Törnqvist (2006). The author contributed the majority of this work.

Paper D: Fast Particle Filters for Multi-Rate Sensors

T. Schön, D. Törnqvist, and F. Gustafsson. Fast particle filters for multi-rate sensors. In *The 15th European Signal Processing Conference (EUSIPCO 2007)*, Poznan, Poland, Sept. 2007b.

Summary: Computational complexity is a major concern for practical use of the versatile particle filter (PF) for nonlinear filtering applications. Previous work to mitigate the inherent complexity includes the marginalized particle filter (MPF), with the FastSLAM algorithm as one important case. MPF utilizes a linear Gaussian sub-structure in the problem, where the Kalman filter (KF) can be applied. While this reduces the state dimension in the PF, the present work aims at reducing the sampling rate of the PF. The algorithm is derived for a class of models with linear Gaussian dynamic model and two multi-rate sensors, with different sampling rates, one slow with a nonlinear and/or non-Gaussian measurement relation and one fast with a linear Gaussian measurement relation. For this case, the KF is used to process the information from the fast sensor and the information from the slow sensor is processed using the PF. The problem formulation covers the important special case of fast dynamics and one slow sensor, which appears in many navigation and tracking problems.

Background and contribution: Based on an idea from Fredrik Gustafsson, Thomas Schön and the author of this thesis worked together on the derivations and the simulation example. The author contributed by writing the section on simulations and parts of the other sections.

Paper E: A multiple UAV system for vision-based search and localization

> J. Tisdale, A. Ryan, Z. Kim, D. Törnqvist, and J. K. Hedrick. A multiple UAV system for vision-based search and localization. In *American Control Conference*, Seattle, WA, USA, June 2008.

Summary: The contribution of this paper is an experimentally verified real-time algorithm for combined probabilistic search and track using multiple unmanned aerial vehicles (UAVs). Distributed data fusion provides a framework for multiple sensors to search for a target and accurately estimate its position. Vision based sensing is employed, using fixed downward-looking cameras. These sensors are modeled to include vehicle state uncertainty and produce an estimate update regardless of whether the target is detected in the frame or not. This allows for a single framework for searching or tracking, and requires non-linear representations of the target position probability density function (PDF) and the sensor model. While a grid-based system for Bayesian estimation was used for the flight demonstrations, the use of a particle filter solution has also been examined.

Multi-aircraft flight experiments demonstrate vision-based localization of a stationary target with estimated error covariance on the order of meters. This capability for real-time distributed estimation will be a necessary component for future research in information-theoretic control.

Background and contribution: During the summer of 2007, the author visited the Center for Collaborative Control of Unmanned Vehicles (C3UV) at University of California, Berkeley in USA. The group had been working with grid based methods for positioning of a ground target using UAVs with camera. The author of this thesis solved the problem using a PF and wrote the sections about that in this paper.

Other publications

Publications not included in this thesis, but that might be of interest are:

> R. Karlsson, T. B. Schön, D. Törnqvist, G. Conte, and F. Gustafsson. Utilizing model structure for efficient simultaneous localization and mapping for a UAV application. In *Proceedings of IEEE Aerospace Conference*, 2008.
>
> T. Schön, R. Karlsson, D. Törnqvist, and F. Gustafsson. A framework for simultaneous localization and mapping utilizing model structure. In *The 10th International Conference on Information Fusion*, Quebec, Canada, Aug. 2007a.
>
> D. Törnqvist. *Statistical Fault Detection with Applications to IMU Disturbances*. Licentiate thesis no. 1258, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, June 2006.
>
> D. Törnqvist and F. Gustafsson. Eliminating the initial state for the generalized likelihood ratio test. In *Proceedings of IFAC Symposium SAFEPROCESS*, Beijing, China, Aug. 2006.
>
> D. Törnqvist, F. Gustafsson, and I. Klein. GLR tests for fault detection over sliding data windows. In *Proceedings of the 16th IFAC World Congress*, Prague, Czech Republic, July 2005.

Part I

Background Theory

2

Modeling

AS HUMANS, WE INTERACT with systems in our every day life. We drive our cars, ride our bikes, control our body movements and so on. All of this is done without using any mathematics at all. We are using *mental* models built by our brain from previous experiences. To make a robot interact with a system or understand its sensor measurements, mental models are hard to use. Therefore, *mathematical* models are derived that can easily be used by a computer. Using this type of models, the robot can control systems, relate to measurements, and even predict the future.

This chapter is divided into two parts. The first part discusses general modeling and the second part derives models for orientation and position of a rigid body.

2.1 System Descriptions

The behavior of most physical systems can be mathematically modeled using differential equations. These equations can be in different standard forms. This section will primarily focus on state space models, also extended to the case where data batches are treated. An efficient parametrization of faults over a data batch is also presented.

2.1.1 State Space Model

A fairly general description of a physical system is provided by the nonlinear state space model

$$\dot{x} = f(x, u, f, v, t), \tag{2.1a}$$

$$y = h(x, u, f, e, t), \tag{2.1b}$$

where x is the state of the system, u is the input signal, f is a fault, v is process noise,e measurement noise, and y is the measured signal. All variables are assumed to be time-varying. The functions $f(\,\cdot\,)$ and $h(\,\cdot\,)$ are in general nonlinear functions and this class of systems is for example discussed in the textbooks Khalil (2002) and Isidori (1995). The class of models where $f(\,\cdot\,)$ and $h(\,\cdot\,)$ are linear functions is called *linear systems*. Linear systems are in general easier to handle and hence there exist more powerful results for them, see for instance Rugh (1996) and Kailath et al. (2000). To be able to use linear theory also for nonlinear systems, the nonlinear systems must be approximated with a linear model.

Linear Model

A time-varying linear state space model can be written as

$$\dot{x} = A(t)x + B^u(t)u + B^f(t)f + B^v(t)v, \tag{2.2a}$$

$$y = C(t)x + D^u(t)u + D^f(t)f + e. \tag{2.2b}$$

where t is used to denote time.

Discrete-Time Models

A computer can only handle discrete-time data, so measurements will be sampled in time. Thus, the system model has to be sampled in order to describe the measurements. A general nonlinear time-discrete model can be described as

$$x_t = f_{t-1}(x_{t-1}, u_{t-1}, f_{t-1}, v_{t-1}), \tag{2.3a}$$

$$y_t = h_t(x_t, u_t, f_t, e_t). \tag{2.3b}$$

For time-discrete systems, t will be used to denote the sample number rather than time.

Sampling of linear systems is described in Rugh (1996). In the case of a piecewise constant time-invariant linear system with piecewise constant input signal, the matrices of the sampled system can be computed as

$$F_t = e^{A(t)T}, \tag{2.4a}$$

$$G_t^s = \int\limits_0^T e^{A(t)\tau}\, d\tau B^s(t), \tag{2.4b}$$

where $s \in \{u, f, v\}$ and T is the sampling time. The discrete-time system can then be written as

$$x_t = F_{t-1}x_{t-1} + G_{t-1}^u u_{t-1} + G_{t-1}^f f_{t-1} + G_{t-1}^v v_{t-1}, \tag{2.5a}$$

$$y_t = H_t x_t + H_t^u u_t + H_t^f f_t + e_t. \tag{2.5b}$$

2.1.2 Batched Systems

It is sometimes convenient to describe the behavior of a system over a certain time window. The system can then be described in batched form, directly derived from the linear state-space form. The batch form is often used in fault detection and diagnosis, see Chow and Willsky (1984), Gertler (1998), Gustafsson (2001), Paper B and C. Stack L signal values to define signal vectors like $\mathbb{Y} = \left(y_{t-L+1}^T, \dots, y_t^T\right)^T$, for all signals. If the initial state in a time window of length L is known, the outputs in that window can be computed as

$$\mathbb{Y} = \mathcal{O}_t x_{t-L+1} + \bar{H}_t^u \mathbb{U} + \bar{H}_t^f \mathbb{F} + \bar{H}_t^v \mathbb{V} + \mathbb{E}, \tag{2.6}$$

with the extended observability matrix

$$\mathcal{O}_t = \begin{pmatrix} H_{t-L+1} \\ H_{t-L+2} F_{t-L+1} \\ \vdots \\ H_t \prod_{k=t-1}^{t-L+1} F_k \end{pmatrix}, \tag{2.7}$$

which in the time invariant case will have the following form

$$\mathcal{O} = \begin{pmatrix} H \\ HF \\ \vdots \\ HF^{L-1} \end{pmatrix}. \tag{2.8}$$

The matrices determining how the input signals affect the system are described by

$$\bar{H}_t^s = \begin{pmatrix} H_{t-L+1}^s & 0 & \cdots & 0 \\ H_{t-L+2} G_{t-L+1}^s & H_{t-L+2}^s & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ H_t \prod_{k=t-1}^{t-L+2} F_k G_{t-L+1}^s & \cdots & H_t G_{t-1}^s & H_t^s \end{pmatrix}, \tag{2.9}$$

and in case of a time invariant system, they will have the form of a Toeplitz matrix

$$\bar{H}^s = \begin{pmatrix} H^s & 0 & \cdots & 0 \\ HG^s & H^s & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ HF^{L-2} G^s & \cdots & HG^s & H^s \end{pmatrix}. \tag{2.10}$$

It is often convenient to describe a model where the influence of the input is removed. Therefore define

$$\mathbb{Z} \triangleq \mathbb{Y} - \bar{H}_t^u \mathbb{U} = \mathcal{O}_t x_{t-L+1} + \bar{H}_t^f \mathbb{F} + \bar{H}_t^v \mathbb{V} + \mathbb{E}. \tag{2.11}$$

2.1.3 Structured Faults

The influence of a fault is often correlated in time and is changing much slower than white noise. It is therefore natural to model the behavior of the fault with as few parameters as possible. The fault can be parameterized using an orthogonal basis generated by the Chebyshev polynomials. The Chebyshev polynomials are orthogonal polynomials of a discrete variable, see Abramowitz and Stegun (1965), Rivlin (1974). The polynomials, denoted $\Phi_n(t)$, are orthogonal in the interval $0 \leq t \leq L-1$, where t is an integer. Thus, the vector of length L

$$\Phi_n^T = \begin{pmatrix} \Phi_n(0) \\ \vdots \\ \Phi_n(L-1) \end{pmatrix}, \tag{2.12}$$

is orthogonal to another vector Φ_m when $n \neq m$. The Chebyshev polynomials are defined by the function

$$\Phi_n(t) = n!\Delta^n \left[\binom{t}{n} \binom{t-L}{n} \right], \tag{2.13}$$

where Δ^n is the difference operator defined as

$$\Delta f(t) = f(t+1) - f(t), \quad \Delta^{n+1} f(t) = \Delta(\Delta^n f(t)). \tag{2.14}$$

The orthonormal vectors Φ_n will be used as basis vectors for the fault. Denoting the parameters θ_t, the fault vector is modeled as

$$\mathbb{F} = \Phi^T \theta_t, \tag{2.15}$$

where

$$\Phi^T = \begin{pmatrix} \Phi_1^T & \Phi_2^T & \cdots \end{pmatrix}. \tag{2.16}$$

The regressors obtained have the form of a constant, linear, quadratic term and so on. This means that a step fault can be described using a one dimensional basis and smooth faults using a low-dimensional basis. An example of a parameterized fault is given in Example 2.1.

Example 2.1: Parameterized Fault

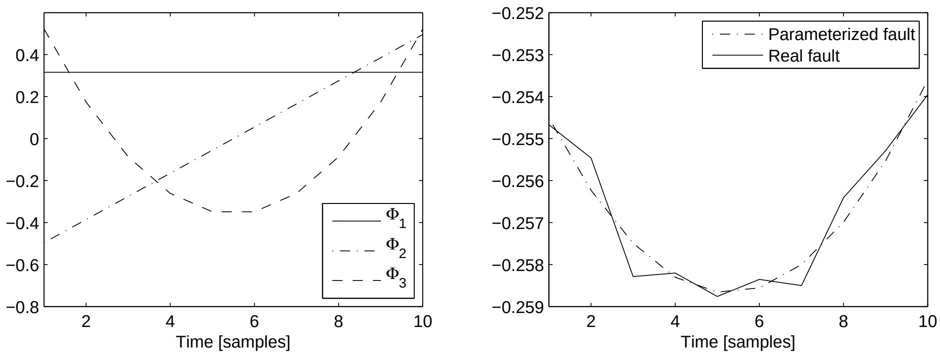
The output from a magnetometer (compass) is monitored. After some time, the magnetic field is disturbed by a magnetic object. This disturbance can be modeled using the Chebyshev polynomial matrix Φ^T. To estimate the model parameters, the following loss function is minimized

$$\hat{\theta}_t = \arg\min_{\theta_t} \|\mathbb{F} - \Phi^T \theta_t\|^2. \tag{2.17}$$

The fault is parameterized using three basis functions shown in Figure 2.1a. The parameterized fault is shown in Figure 2.1b.

2.2 Rigid Body Modeling

In many situations, the orientation and position of a system are of interest. Navigation or control of cars, airplanes, and robots are only a few of the examples. For modeling of ground applications it is often enough to model the two-dimensional position and a heading angle, while airborne applications need three-dimensional position and orientation. This chapter will cover the latter, both modeling the dynamics and also measurements from an Inertial Measurement Unit, IMU. The IMU modeled in this section not only consists of the traditional accelerometer and gyroscope, but also a magnetometer (compass). The results derived in this section are used for modeling of a UAV in Paper A.



(a) *Fault basis* ***(b)*** *Parameterized fault*

Figure 2.1: *The basis functions (a) are used to parameterize the fault from a magnetometer disturbance (b).*

2.2.1 Coordinate Systems

To describe a moving body system, two coordinate systems are used. One system, that is fixed to the world (earth), referred to as the W-system, and one that is following the moving body, referred to as the B-system. Figure 2.2 illustrates the coordinate systems. Theoretically, the measurements from the IMU are made in an inertial frame. The rotation of the Earth introduces Coriolis forces, but since these effects are small, they are neglected here. This allows us to use the W-system as the home of the IMU measurements. The relation between the coordinate systems is described in two steps. First, the translation from the origin of the W-system to the origin of the B-system is described with the vector p. Second, the orientation from the W-system to the B-system is described by the unit quaternion q_{bw} or the corresponding rotation matrix $R(q_{bw})$. Unit quaternions can be used to describe orientation numerically more robust than, for example, Euler angles. For more information about quaternions, see Appendix B.

2.2.2 Dynamic Model

The translation between the W-system and the B-system p is also called the *position* of the body. The dynamic model needs to have the position p and the rotation q_{bw} as states. Since the accelerations of the B-system are measured, these have to be modeled as states as well as the velocity v^w. The state vector can then be chosen as

$$x = \begin{pmatrix} \mathsf{p}^{wT} & \mathsf{v}^{wT} & \mathsf{a}^{wT} & {q_{bw}}^T \end{pmatrix}^T, \tag{2.18}$$

where p^w is translation from the W-system to the B-system (described in the W-system), v^w is the velocity vector of the B-system relative to the W-system (described in the W-system), a^w is acceleration of the B-system relative to the W-system (described in the W-system), q_{bw} is the unit quaternion describing the orientation of the B-system.

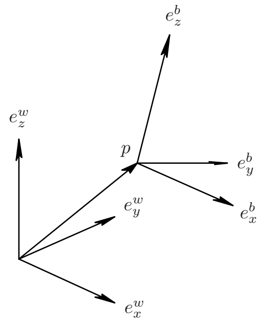


Figure 2.2: *The two coordinate systems describing the moving body system. The coordinate system* (e_x^b, e_y^b, e_z^b) *is moving with the body and the coordinate system* (e_x^w, e_y^w, e_z^w) *is fixed.*

The dynamics of the translation is easy, namely

$$\begin{aligned}\dot{\mathsf{p}}^w &= \mathsf{v}^w,\\ \dot{\mathsf{v}}^w &= \mathsf{a}^w.\end{aligned}$$

Since nothing is known about the derivative of the acceleration, this is modeled as noise,

$$\dot{\mathsf{a}}^w = v_1,$$

where v_1 is process noise. The dynamics of the unit quaternion q_{bw} is derived in Appendix B and the derivative is given in (B.36). The result is

$$\dot{q}_{bw} = \frac{1}{2} S(\omega_{bw}^b) q_{bw}, \tag{2.19}$$

where

$$S(\omega_{bw}^b) = \begin{pmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{pmatrix}, \tag{2.20}$$

and $\omega_{bw}^b = (\omega_x \; \omega_y \; \omega_z)^T$ denotes the angular velocity of the B-system relative to the W-system described in the B-system. The angular velocity ω_{bw}^b is measured by the gyroscopes and can therefore be modeled as a state which is then modeled as a measurement. However, to reduce the dimension of the state space and avoid the nonlinearity which would otherwise have been introduced, the measurement of the angular velocity is used as a known parameter in the dynamics. This approach would only be theoretically correct if ω_{bw}^b could be measured without noise. However, since the measurement noise from the gyroscopes v_2 is small, the approximation used here is to use $\omega_{bw}^b + v_2$ as angular velocity. Thus, the dynamics is approximated as

$$\dot{q}_{bw} \approx \frac{1}{2} S(\omega_{bw}^b + v_2) q_{bw}. \tag{2.21}$$

Using (B.36), it is possible to write

$$S(\omega_{bw}^b + v_2)q_{bw} = S(\omega_{bw}^b)q_{bw} + S'(q_{bw})v_2. \tag{2.22}$$

where

$$S'(q_{bw}) \begin{pmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{pmatrix}, \tag{2.23}$$

and $q_{bw} = (q_0\ q_1\ q_2\ q_3)^T$. Then, (2.21) can be written as

$$\dot{q}_{bw} \approx \frac{1}{2}S(\omega_{bw}^b)q_{bw} + \frac{1}{2}S'(q_{bw})v_2. \tag{2.24}$$

Thus, the total dynamic model is given by

$$\begin{pmatrix} \dot{\mathsf{p}}^w \\ \dot{\mathsf{v}}^w \\ \dot{\mathsf{a}}^w \\ \dot{q}_{bw} \end{pmatrix} = \underbrace{\begin{pmatrix} 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2}S(\omega_{bw}^b) \end{pmatrix}}_{A} \begin{pmatrix} \mathsf{p}^w \\ \mathsf{v}^w \\ \mathsf{a}^w \\ q_{bw} \end{pmatrix} + \underbrace{\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ I & 0 \\ 0 & \frac{1}{2}S'(q_{bw}) \end{pmatrix}}_{B} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}. \tag{2.25}$$

2.2.3 Measurements

The sensors that are available in the system are gyroscopes, accelerometers and magnetometer (compass). The measurements from the gyroscopes are, as discussed above, directly incorporated in the dynamics and are therefore not described in the measurement equations.

The accelerometers measure both the free acceleration of the body and the earth gravitational field described in the B-system. The measurement equation can therefore be written as

$$y_{a,t} = \mathsf{a}_t^b - g^b + e_{a,t} = \underbrace{R(q_{bw,t})(\mathsf{a}_t^w - g^w)}_{\triangleq h_a(q_{bw,t},\mathsf{a}_t^w)} + e_{a,t}, \tag{2.26}$$

where $R(\,\cdot\,)$ is the rotation matrix given in (B.31), a is the free acceleration, g is the gravity vector and e_a is the measurement noise.

The measurement from the magnetometer is a normalized vector $\bar{n}_{np}^b$ given in the B-system and pointing along the earth magnetic field. When using an ordinary compass, the reading will be two dimensional giving the heading toward the magnetic north pole. Since this measurement is three dimensional, information is also given about the magnetic inclination. The inclination, also called *dip angle*, is the angle between the earth's tangent plane and the magnetic field vector. The magnetic inclination depends on the position on earth and is $0°$ at the magnetic equator and $90°$ at the magnetic poles. The inclination in Linköping, Sweden, is about $70°$. The heading information will of course be less accurate when the inclination approaches $90°$. For more information about the earth magnetic field, see National Geomagnetism Program.

Since the earth magnetic field is very weak, it can easily be disturbed by an electric motor or ferromagnetic objects such as beams or other iron constructions. The disturbance can be modeled as an additive signal, here denoted $d_{m,t}$. The measurement equation can thus be written as

$$y_{m,t} = \bar{n}^b_{np,t} + d_{m,t} + e_{m,t} = \underbrace{R(q_{bw,t})\bar{n}^w_{np}}_{\triangleq h_m(q_{bw,t})} + d_{m,t} + e_m. \tag{2.27}$$

The total measurement equation becomes

$$y_t = \begin{pmatrix} y_{a,t} \\ y_{m,t} \end{pmatrix} = h(q_{bw,t}, \mathsf{a}^w_t) + \begin{pmatrix} 0 \\ I \end{pmatrix} d_{m,t} + e_t, \tag{2.28}$$

which is in the standard form given in Section 2.1.

2.2.4 Discrete-Time Dynamic Model

The dynamic model (2.25) is divided into two parts. One containing the translational dynamics of the B-system and the other part the orientation dynamics. The first part contains the states p^w, v^w and a^w and the second part q_{bw}. To simplify the notation and stress the time dependency, q_{bw} is denoted q_t and ω^b_{bw} is denoted ω_t from now on.

Translational Dynamics

For the translational part of the system, the noise inputs to the system are assumed to be constant between the sampling intervals. Then the discrete time system can be written as

$$\begin{pmatrix} \mathsf{p}^w_t \\ \mathsf{v}^w_t \\ \mathsf{a}^w_t \end{pmatrix} = F \begin{pmatrix} \mathsf{p}^w_{t-1} \\ \mathsf{v}^w_{t-1} \\ \mathsf{a}^w_{t-1} \end{pmatrix} + G^v v_{t-1}, \tag{2.29}$$

where

$$F = e^{AT} \quad \text{and} \quad G^v = \int_0^T e^{At} B \, dt.$$

The matrices in (2.29) can be evaluated to

$$F = \begin{pmatrix} I & TI & \frac{T^2}{2}I \\ 0 & I & TI \\ 0 & 0 & I \end{pmatrix} \quad \text{and} \quad G^v = \begin{pmatrix} \frac{T^3}{6}I \\ \frac{T^2}{2}I \\ TI \end{pmatrix}.$$

Attitude Dynamics

The part of the system (2.25) describing the orientation is the quaternion q_t. Assume that the angular velocity ω_t is constant during the sampling intervals. Making this assumption, the orientation dynamics becomes time invariant during the sampling interval. Hence, considering the noise-free dynamic model (2.19), the discrete time model is given by

$$q_t = e^{\frac{1}{2}S(\omega_{t-1})T} q_{t-1}. \tag{2.30}$$

To simplify calculations, the term $e^{\frac{1}{2}S(\omega_t)T}$ has to be rewritten. It can be rewritten as the series expansion

$$e^{\frac{1}{2}S(\omega_t)T} = \sum_{n=0}^{\infty} \frac{\left(\frac{1}{2}S(\omega_t)T\right)^n}{n!} = \sum_{n=0}^{\infty} \left(\frac{T}{2}\right)^n \frac{1}{n!} S(\omega_t)^n. \tag{2.31}$$

The matrix $S(\omega_t)$ has an interesting property, namely

$$S(\omega_t)^2 = -\|\omega_t\|^2 I. \tag{2.32}$$

Using this result and the series expansion for the cosine, the even part of the series expansion (2.31) can be written as

$$\sum_{n=0}^{\infty} \left(\frac{T}{2}\right)^{2n} \frac{1}{(2n)!}(-1)^n \|\omega_t\|^{2n} = \cos\left(\frac{\|\omega_t\|T}{2}\right) I \tag{2.33}$$

Using knowledge about the series expansion of the sine, the odd part can be written as

$$\begin{aligned} &\sum_{n=0}^{\infty} \left(\frac{T}{2}\right)^{2n+1} \frac{1}{(2n+1)!} S(\omega_t)^{2n+1} \\ &\qquad = \frac{T}{2} S(\omega_t) \sum_{n=0}^{\infty} \left(\frac{T}{2}\right)^{2n} \frac{1}{(2n+1)!}(-1)^n \frac{\|\omega_t\|^{2n+1}}{\|\omega_t\|} \\ &\qquad\qquad = \frac{T}{2} S(\omega_t) \frac{\sin\left(\|\omega_t\|T/2\right)}{\|\omega_t\|}. \end{aligned} \tag{2.34}$$

Combining the odd and even parts of the series expansion, the unit quaternion dynamics can be written as

$$q_t = \left(\cos\left(\frac{\|\omega_{t-1}\|T}{2}\right) I + \frac{\sin\left(\frac{\|\omega_{t-1}\|T}{2}\right)}{\|\omega_{t-1}\|} S(\omega_{t-1}) \right) q_{t-1}. \tag{2.35}$$

If the sample rate is high enough so that the product $\|\omega_{t-1}\|T$ is small, the small angle approximation can be used to simplify the dynamic equation to

$$q_t = \left(I + \frac{T}{2} S(\omega_{t-1})\right) q_{t-1}. \tag{2.36}$$

Now, reintroducing the measurement noise on ω_{t-1}, (2.22) can be used to derive the following expression

$$q_t = \left(I + \frac{T}{2} S(\omega_{t-1})\right) q_{t-1} + \frac{T}{2} S'(q_{t-1}) v_{2,t-1}. \tag{2.37}$$

2.2.5 Linearization

To do estimation using linear filtering theory or to use linear theory for fault detection, the system must be linearized. The dynamic equations (2.29) and (2.37) are already linear time-varying equations, but the measurement equation is nonlinear. It is therefore linearized with a first order Taylor series. Starting with the accelerometer part of the measurement equation, it can be approximated as

$$y_{a,t} = h_a(q_t) + e_{a,t} \approx h_a(\hat{q}_{t|t-1}) + H_{a,t}(q_t - \hat{q}_{t|t-1}) + e_{a,t}, \tag{2.38}$$

where:

$$H_{a,t} = \left.\frac{\partial h_a(q)}{\partial q}\right|_{q=\hat{q}_{t|t-1}} \tag{2.39}$$

The magnetometer part of the measurement equation is approximated as

$$\begin{aligned} y_{m,t} &= h_m(q_t, \mathsf{a}^w_t) + e_{m,t} \\ &\approx h_m(\hat{q}_{t|t-1}) + H^m_t(q_t - \hat{q}_{t|t-1}) + d_{m,t} + e_{m,t}, \end{aligned} \tag{2.40}$$

where:

$$H^m_t = \left.\frac{\partial h_m(q)}{\partial q}\right|_{q=\hat{q}_{t|t-1}}$$

To form the linearized system, a new measurement variable is computed with known information as

$$\tilde{y}_t \triangleq y_t - \begin{pmatrix} h_a(\hat{q}_{t|t-1}) \\ h_m(\hat{q}_{t|t-1}) \end{pmatrix} + \begin{pmatrix} H^a_t \\ H^m_t \end{pmatrix} \hat{q}_{t|t-1} = \begin{pmatrix} H^a_t \\ H^m_t \end{pmatrix} q_t + \begin{pmatrix} 0 \\ I \end{pmatrix} d_{m,t} + e_t, \tag{2.41}$$

where the Jacobians in the measurement equation (2.44b) are

$$H^a_t = 2g \begin{pmatrix} -q_2 & q_3 & -q_0 & q_1 \\ q_1 & q_0 & q_3 & q_2 \\ q_0 & -q_1 & -q_2 & q_3 \end{pmatrix} \tag{2.42}$$

and

$$H^{mT}_t = 2 \begin{pmatrix} n_x q_0 + n_y q_3 - n_z q_2 & -n_x q_3 + n_y q_0 + n_z q_1 & n_x q_2 - n_y q_1 + n_z q_0 \\ n_x q_1 + n_y q_2 + n_z q_3 & n_x q_2 - n_y q_1 + n_z q_0 & n_x q_3 - n_y q_0 - n_z q_1 \\ -n_x q_2 + n_y q_1 - n_z q_0 & n_x q_1 + n_y q_2 + n_z q_3 & n_x q_0 + n_y q_3 - n_z q_2 \\ -n_x q_3 + n_y q_0 + n_z q_1 & -n_x q_0 - n_y q_3 + n_z q_2 & n_x q_1 + n_y q_2 + n_z q_3 \end{pmatrix}, \tag{2.43}$$

where the magnetic vector has the following coordinates $\bar{n}^f_{np} = (n_x \; n_y \; n_z)^T$. Then the linearized system can now be written as

$$q_t = F_{t-1} q_{t-1} + G^v_{t-1} v_{t-1}, \tag{2.44a}$$

$$\tilde{y}_t = \underbrace{\begin{pmatrix} H^a_t \\ H^m_t \end{pmatrix}}_{\triangleq H_t} q_t + \underbrace{\begin{pmatrix} 0 \\ I \end{pmatrix}}_{\triangleq H^d_t} d_{m,t} + e_t. \tag{2.44b}$$

We have now derived an approximative linear model on the form (2.5) of a nonlinear model on the form (2.3). This linearized model will be used in combination with the filtering and detection methods in Chapter 3.

3

Estimation and Detection Theory

USING MEASUREMENTS to gain knowledge about the state of a system is often referred to as estimation or statistical inference. This is a well studied area and many methods used today are based on theory developed in the 18th century. This chapter will start with a historical overview over the debate about statistical viewpoints that still goes on today. Section 3.2 discusses some of the common parameter estimation methods. Taking the Bayesian view on recursive estimation in Section 3.3, this is then used to derive the Kalman filter, extended Kalman filter, particle filter and the Rao-Blackwellized particle filter in Section 3.4, 3.5, 3.6, and 3.7 respectively.

Another area, closely related to estimation, is detection. To detect if a change has occurred in a system, hypotheses are formed and statistical tests are used to find the most likely hypothesis. This is the subject of Section 3.8.

3.1 Bayesians and Frequentists

The measurement model describes how a measurement y relates to the system state x, often in a probabilistic way as $p(y|x)$. The statistical inference problem is sometimes referred to as the problem of inverse probability. The question is if it is possible to use the model and gathered measurements to compute the inverse probability, i.e., $p(x|y)$. The British mathematician and minister Thomas Bayes treated the statistical inference problem already in the 18th century. His publication (Bayes, 1763) introduced the theorem that is referred to as Bayes' theorem

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}. \tag{3.1}$$

This theorem allows us to compute the inverse probability given the *prior probability* $p(x)$, which can be more or less informative. The *Bayesian view* of the world is through the so called *degree of belief*. The belief should reflect the uncertainty related to the system state and is represented as a probability distribution. The Bayesians believe that only the data is real, so given the data and the prior belief a posterior belief can be calculated.

Mathematicians started to object against the Bayesian world-view during the late 19th century and in the 1920s Sir Ronald Aylmer Fisher founded a new view of statistics with Fisher (1922). The arguments against the Bayesian view were that prior beliefs could not be accepted in an objective world-view. Fisher's view was that there is a true system state x which is deterministic, but unknown. The probability of an outcome should be seen as the relative frequency for the outcome, given data from a long-run experiment. A person having this view of statistics is therefore often referred to as a *frequentist*.

Bayesians and frequentists have different approaches to the statistical inference problem, even though the solutions often are very similar. One example is the Kalman filter (KF), see Section 3.4. The frequentist views the KF as a recursive minimum variance least squares estimator. They are able to prove that it is the best linear unbiased estimator, BLUE, but there might be a better nonlinear estimator. The Bayesians on the other hand starts with a prior belief of the state distribution. This belief is then recursively propagated over time using Bayes' theorem and the solution for a linear/Gaussian system turns out to be the KF. In this case the filters are identical, but the interpretation of the solution is different.

The debate among Bayesians and frequentists has been ongoing since the days of Fisher and a historical story is given in Howie (2002). The Bayesian view has regained its popularity for use with nonlinear filtering using Monte Carlo methods. Lately, many books promoting this view has been published, e.g., Jaynes (2003), Robert (2001), Šmídl and Quinn (2006). It cannot be said that one view always is better than the other, but there have been comparative studies, see for instance Samaniego and Reneau (1994).

3.2 Parameter Estimation

The parameter estimation problem deals with finding the "best" parameters from a batch of measurements $y_{1:t}$. The relation between the parameters and the measurements is usually in the following form

$$y_t = h(x) + e_t, \tag{3.2}$$

where x are the parameters to be estimated and e_t is the measurement noise. What is meant by best can of course vary depending on the situation and can be defined by minimizing or maximizing a cost function. There are several cost functions resulting in different estimation methods. Some of these are

- The *Least Squares* (LS) method finds the parameters that gives the smallest squared error

$$\hat{x} = \arg\min_x \sum_t \|y_t - h(x)\|_2^2. \tag{3.3}$$

- The *Maximum likelihood* (ML) estimator computes the parameters that are most likely to produce the measurements, the cost function is

$$\hat{x} = \arg\max_x p(y_{1:t}|x) \tag{3.4}$$

- The *Maximum a posteriori* (MAP) estimator instead computes the most likely parameters given the measurements
$$\hat{x} = \arg\max_{x} p(x|y_{1:t}). \tag{3.5}$$
Note that the relation to ML is given by Bayes' rule (3.1), where it is clear that the prior $p(x)$ will affect the solution.

A special case of parameter estimation that will be important in the included papers about detection, Paper B and C, is estimation of the initial state in a given time window. Section 2.1.2 gives the following model of the outputs in a time window

$$\mathbb{Z} = \mathcal{O}_t x_{t-L+1} + \bar{H}_t^v \mathbb{V} + \mathbb{E} \tag{3.6}$$

This type of model is known as the *general Gauss-Markov linear model*. The parameters in x_{t-L+1} can be estimated with minimum variance by solving the generalized least squares problem

$$\hat{x}_{t-L+1} = \arg\min_{x_{t-L+1}} (\mathbb{Z} - \mathcal{O}_t x_{t-L+1})^T S^{-1} (\mathbb{Z} - \mathcal{O}_t x_{t-L+1}) \tag{3.7}$$

where the noise distribution is assumed to be Gaussian with $S = \mathsf{Cov}(\bar{H}_t^v \mathbb{V} + \mathbb{E})$, see Björck (1996). This can be rewritten as the least squares problem

$$\hat{x}_{t-L+1} = \arg\min_{x_{t-L+1}} \|S^{-1/2}\mathbb{Z} - S^{-1/2}\mathcal{O}_t x_{t-L+1}\|_2^2, \tag{3.8}$$

which has the solution

$$\hat{x}_{t-L+1} = (S^{-1/2}\mathcal{O}_t)^\dagger S^{-1/2}\mathbb{Z}, \tag{3.9}$$

where † denotes the Moore-Penrose pseudo inverse, see Golub and van Loan (1996). The estimate will be distributed according to

$$\hat{x}_{t-L+1} = (S^{-1/2}\mathcal{O}_t)^\dagger S^{-1/2}\mathbb{Z} \sim \mathsf{N}(x_{t-L+1}, (\mathcal{O}_t^T S^{-1} \mathcal{O}_t)^{-1}), \tag{3.10}$$

where N denotes the multivariate Gaussian distribution, see Appendix A.1.

3.3 Recursive Bayesian Filtering

This section will take the Bayesian approach to recursive filtering. An initial distribution is available which is then propagated through the filter using Bayes' rule and the system model. The model used in this section is fairly general. Consider the following Hidden Markov Model (HMM), given by the densities

$$p(x_{t+1}|x_t), \tag{3.11a}$$
$$p(y_t|x_t), \tag{3.11b}$$

with initial state distribution

$$p(x_0). \tag{3.11c}$$

The hidden parameter is the state x_t, which is observed through the measurement y_t. The model for the hidden parameters or states (3.11a) is a Markov process which implies that the future states are independent of old states given the current state x_t, i.e., $p(x_t|x_{1:t-1}) = p(x_t|x_{t-1})$. The interpretation of this is that all information needed for prediction is contained in the most current state x_t. This type of model includes both nonlinear and non-Gaussian models.

The filtering problem asks for the probability of the current state given the measurements, that is, the marginal filtering distribution

$$p(x_t|y_{1:t}). \tag{3.12}$$

This is also referred to as the *posterior* distribution since it is conditioned on the last measurement y_t. To form a recursive solution using only the known model (3.11), Bayes' rule is applied,

$$\begin{aligned} p(x_t|y_{1:t}) &= p(x_t|y_t, y_{1:t-1}) \\ &= \frac{p(y_t|x_t, y_{1:t-1})p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} = \frac{\overbrace{p(y_t|x_t)}^{(3.11b)}\,p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}, \end{aligned} \tag{3.13}$$

where the time update or *a priori* distribution is calculated as

$$p(x_t|y_{1:t-1}) = \int \underbrace{p(x_t|x_{t-1})}_{(3.11a)}\underbrace{p(x_{t-1}|y_{1:t-1})}_{\text{Previous (3.13)}}\, dx_{t-1}, \tag{3.14}$$

the normalization factor is

$$p(y_t|y_{1:t-1}) = \int \underbrace{p(y_t|x_t)}_{(3.11b)}\underbrace{p(x_t|y_{1:t-1})}_{(3.14)}\, dx_t. \tag{3.15}$$

The analytical solution to the filtering problem above is only possible to compute in a few special cases, e.g., the KF is the solution for linear systems with Gaussian noise. For a more general case, numerical methods can be used to solve the problem approximately. There are for example grid-based methods and the particle filter (PF).

Another related problem is to estimate the posterior distribution for the entire trajectory $x_{1:t}$. This would also provide a solution to the smoothing problem, i.e., finding $p(x_n|y_{1:t})$ where $n < t$. The recursive solution to this problem, using Bayes' rule, would be

$$p(x_{1:t}|y_{1:t}) = \frac{p(y_t|x_t)p(x_{1:t}|y_{1:t-1})}{p(y_t|y_{1:t-1})}, \tag{3.16}$$

where

$$p(x_{1:t}|y_{1:t-1}) = p(x_t|x_{t-1})p(x_{1:t-1}|y_{1:t-1}), \tag{3.17}$$

$$p(y_t|y_{1:t-1}) = \int p(y_t|x_t)p(x_{1:t}|y_{1:t-1})\, dx_{1:t}. \tag{3.18}$$

This problem formulation is often used in particle filtering and this will be further discussed in Section 3.6.2.

3.4 Kalman Filter

The KF is the solution to the recursive Bayesian filtering problem described in Section 3.3, given that the system model is linear and the involved noise distributions are Gaussian. The KF can also be viewed from the frequentist view as discussed in Section 3.1. With this view it can be shown that the KF is the Best Linear Unbiased Estimator (BLUE), which means that its estimate is unbiased and the estimation error has least covariance among the class of linear and unbiased estimators. The filter was first presented in Kalman (1960) and is a commonly used estimator. Many books give a detailed description of the KF, see for example Kailath et al. (2000) and Anderson and Moore (1979).

The filter will be derived here using the Bayesian problem formulation and is summarized in Algorithm 3.1. The derivations are made without going into detail which is straightforward but lengthy. The interested reader is referred to Ho and Lee (1964), Nordlund (2002) or Schön (2006) for the complete derivations. To fit the framework, the linear models from Chapter 2 are formulated in terms of the following Probability Density Functions (PDFs)

$$p(x_t|x_{t-1}) = \mathsf{N}(x_t; F_t x_{t-1} + G_t^u u_t, G_t^v Q_t G_t^{vT}), \tag{3.19a}$$

$$p(y_t|x_t) = \mathsf{N}(y_t; H_t x_t + H_t^u u_t, R_t). \tag{3.19b}$$

Initialization

The initial distribution is Gaussian and parameterized as

$$p(x_0) = \mathsf{N}(x_0; \hat{x}_{0|0}, P_{0|0}) \tag{3.20}$$

Time Update

The time update corresponds to making a one-step prediction according to the system model. The prediction is calculated by the integral in (3.14), which in this case can be computed analytically due to the linear and Gaussian structure. The result is again a Gaussian distribution

$$p(x_t|y_{1:t-1}) = \int p(x_t|x_{t-1}) p(x_{t-1}|y_{1:t-1})\, dx_{t-1} = \mathsf{N}(x_t; \hat{x}_{t|t-1}, p_{t|t-1}), \tag{3.21a}$$

where

$$\hat{x}_{t|t-1} = F_{t-1}\hat{x}_{t-1|t-1} + G_{t-1}^u u_{t-1}, \tag{3.21b}$$

$$P_{t|t-1} = F_{t-1} P_{t-1|t-1} F_{t-1}^T + G_{t-1}^v Q_{t-1} {G_{t-1}^v}^T. \tag{3.21c}$$

Measurement Update

The measurement update will infer the measurement to the distribution. This is accomplished by completing the recursion in (3.13) which includes computing the integral in (3.15). Again, this can be done analytically due to the linear and Gaussian structure. The end result is

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} = \frac{p(y_t, x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})} = \mathsf{N}(x_t; \hat{x}_{t|t}, P_{t|t}) \tag{3.22a}$$

where

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - H_t\hat{x}_{t|t-1} - H_t^u u_t), \tag{3.22b}$$

$$P_{t|t} = (I - K_t H_t)P_{t|t-1}, \tag{3.22c}$$

$$K_t = P_{t|t-1}H_t^T(H_t P_{t|t-1}H_t^T + R_t)^{-1}. \tag{3.22d}$$

This completes the recursion, the algorithm is summarized in Algorithm 3.1.

Algorithm 3.1 Kalman Filter (KF)

1. Initialize: Set $\hat{x}_{0|0} = x_0$ and $P_{0|0} = P_0$.
2. Time update:

$$\hat{x}_{t|t-1} = F_{t-1}\hat{x}_{t-1|t-1} + G_{t-1}^u u_{t-1} \tag{3.23a}$$

$$P_{t|t-1} = F_{t-1}P_{t-1|t-1}F_{t-1}^T + G_{t-1}^v Q_{t-1} {G_{t-1}^v}^T \tag{3.23b}$$

3. Measurement update:

$$K_t = P_{t|t-1}H_t^T(H_t P_{t|t-1}H_t^T + R_t)^{-1} \tag{3.23c}$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - H_t\hat{x}_{t|t-1} - H_t^u u_t) \tag{3.23d}$$

$$P_{t|t} = (I - K_t H_t)P_{t|t-1} \tag{3.23e}$$

4. Set $t := t + 1$ and repeat from step 2.

3.5 Extended Kalman Filter

To be able to use the KF framework also for a nonlinear system, the model is linearized. Linearizing the system around the estimated trajectory and then applying the KF gives the Extended Kalman Filter, EKF. Given a nonlinear system

$$x_t = f_{t-1}(x_{t-1}, u_{t-1}, v_{t-1}), \tag{3.24a}$$

$$y_t = h_t(x_t, u_t) + e_t, \tag{3.24b}$$

the linear system is a first order Taylor-expansion around the latest state estimate,

$$f(x_t) \approx f(\hat{x}_{t|t}, u_t) + F_t(x_t - \hat{x}_{t|t}), \tag{3.25a}$$

$$g(x_t) \approx g(\hat{x}_{t|t}, u_t) + G_t^v v_t, \tag{3.25b}$$

$$h(x_t) \approx h(\hat{x}_{t|t-1}, u_t) + H_t(x_t - \hat{x}_{t|t-1}), \tag{3.25c}$$

where:

$$F_t \triangleq \left.\frac{\partial f(x,u,v)}{\partial x}\right|_{(x,u,v)=(\hat{x}_{t|t},u_t,0)} \qquad G_t^v \triangleq \left.\frac{\partial f(x,u,v)}{\partial v}\right|_{(x,u,v)=(\hat{x}_{t|t},u_t,0)}$$

$$H_t \triangleq \left.\frac{\partial h(x,u)}{\partial x}\right|_{(x,u)=(\hat{x}_{t|t-1},u_t)}$$

Using this linearization, the state trajectory can be estimated by the EKF in Algorithm 3.2. Observe that optimality is not guaranteed by the EKF, but it often performs good in practice.

Algorithm 3.2 Extended Kalman Filter (EKF)

1. Initialize: Set $\hat{x}_{0|0} = x_0$ and $P_{0|0} = P_0$.
2. Time update:

$$\hat{x}_{t|t-1} = f_{t-1}(\hat{x}_{t-1}, u_{t-1}) \tag{3.26a}$$

$$P_{t|t-1} = F_{t-1}P_{t-1|t-1}F_{t-1}^T + G_{t-1}^v Q_{t-1} {G_{t-1}^v}^T \tag{3.26b}$$

3. Measurement update:

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t\big(y_t - h(\hat{x}_{t|t-1}, u_t)\big) \tag{3.26c}$$

$$P_{t|t} = (I - K_t H_t)P_{t|t-1} \tag{3.26d}$$

$$K_t = P_{t|t-1}H_t^T(H_t P_{t|t-1} H_t^T + R_t)^{-1} \tag{3.26e}$$

4. Set $t := t + 1$ and repeat from step 2.

3.6 Particle Filter

As discussed in Section 3.3, the recursive filtering problem cannot be computed analytically except for a few special cases. The PF solves this problem by using N samples from the desired distribution

$$x_t^{(i)} \sim p(x_t|y_{1:t}), \quad i = 1, \dots, N, \tag{3.27}$$

where $x_t^{(i)}$ denotes the samples or *particles*. The particles can be used to form an approximative distribution as

$$p(x_t|y_{1:t}) \approx \hat{p}(x_t|y_{1:t}) = \sum_{i=1}^{N} q_{t|t}^{i}\delta(x_t - x_t^{(i)}), \quad \sum_{i=1}^{N} q_{t|t}^{(i)} = 1, \quad q_{t|t}^{(i)} \geq 0, \forall i, \tag{3.28}$$

where $\hat{p}(\,\cdot\,)$ denotes an approximative distribution, $\delta(\,\cdot\,)$ is the Dirac delta and $q_{t|t}^{(i)}$ denotes weights dependent on the sampling method used. The Dirac delta $\delta(a)$ can be thought of as a function that is zero except at $a = 0$ where it is infinitely large. Formally, it is a distribution or measure with the following properties

$$\int_{-\infty}^{\infty} \delta(x)\,dx = 1, \qquad \int_{-\infty}^{\infty} f(x)\delta(x-a)\,dx = f(a), \tag{3.29}$$

where the last property is referred to as the sifting property. Having a particle representation of the filtering distribution, the integrals in (3.13) can be solved easily. The PF can be shown to approach the exact solution to the filtering problem if the number of particles goes to infinity, see discussions about convergence results in (Del Moral, 2004) and (Hu et al., 2008). An approximation can be computed with a finite number of particles.

The PF was introduced in Gordon et al. (1993), where the missing key to the PF was found. This was the resampling step that prevented the method to degenerate after a few iterations. The PF stems from the Monte Carlo (MC) integration methods and the initial steps were taken already in the 1940's (Metropolis and Ulam, 1949). Further reading about the PF can be done in the tutorials Arulampalam et al. (2002), Doucet and Johansen (2008), Doucet (1998) and the books Doucet et al. (2001), Ristic et al. (2004).

3.6.1 Derivations

Remember the filtering posterior probability given by (3.13)

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{p(y_t|y_{1:t-1})}. \tag{3.30}$$

The following derivations will follow this decomposition of the filtering distribution.

Initialize

Initialize the PF by sampling from the initial distribution

$$x_0^{(i)} \sim p(x_0), \tag{3.31}$$

and depending on the sampling strategy assign values to $q_{0|0}^{(i)}$. This gives the approximative initial distribution

$$p(x_0) \approx \sum_{i=1}^{N} q_{0|0}^{(i)} \delta(x_0 - x_0^{(i)}). \tag{3.32}$$

Time Update

The time update starts with computing the one-step ahead prediction by using (3.14) as

$$\begin{aligned} p(x_t|y_{1:t-1}) &= \int p(x_t|x_{t-1})p(x_{t-1}|y_{1:t-1})\,dx_{t-1} \\ &\approx \int p(x_t|x_{t-1})\sum_{i=1}^{N} q_{t-1|t-1}^{(i)}\delta(x_{t-1}-x_{t-1}^{(i)})\,dx_{t-1} \\ = \sum_{i=1}^{N} q_{t-1|t-1}^{(i)} \int p(x_t|x_{t-1})\delta(x_{t-1}-x_{t-1}^{(i)})\,dx_{t-1} &= \sum_{i=1}^{N} q_{t-1|t-1}^{(i)} p(x_t|x_{t-1}^{(i)}). \end{aligned} \tag{3.33}$$

This is a continuous distribution, a sum of process noise distributions. To regain a particle distribution, new samples must be generated. This can be done according to

$$x_t^{(i)} \sim \pi(x_t|x_{t-1}^{(i)}, y_t), \quad i = 1, \dots, N, \tag{3.34}$$

where $\pi(\,\cdot\,)$ is a proposal distribution which should be selected so that it has appropriate support. It is often an advantage to include the latest measurement y_t in the proposal distribution as this can improve the numerical properties of the PF (Doucet, 1998). However, in the simplest case, the proposal distribution can be given by the dynamics as $\pi(x_t|x_{t-1}, y_t) = p(x_t|x_{t-1}^{(i)})$. To make the new particle distribution reflect the distribution in (3.33), the weights need to be updated according to

$$q_{t|t-1}^{(i)} \propto \frac{p(x_t^{(i)}|x_{t-1}^{(i)})}{\pi(x_t^{(i)}|x_{t-1}^{(i)}, y_t)} q_{t-1|t-1}^{(i)}, \quad i = 1, \dots, N. \tag{3.35}$$

If the distribution of the prediction should be used for constructing an estimate, normalization to ensure $\sum_i q_{t|t-1}^{(i)} = 1$ is needed. Otherwise, the normalization is taken care of in the measurement update. In the case where the proposal distribution is chosen as $\pi(x_t|x_{t-1}, y_t) = p(x_t|x_{t-1}^{(i)})$ the old weights are simply kept $q_{t|t-1}^{(i)} = q_{t-1|t-1}^{(i)}$. After the time update, the following distribution is available

$$\hat{p}(x_t|y_{1:t-1}) = \sum_{i=1}^{N} q_{t|t-1}^{(i)}\delta(x_t - x_t^{(i)}). \tag{3.36}$$

Measurement Update

The measurement update infers the latest measurement by completing the calculations in (3.30). Observe that the denominator is only a normalizing constant since it is independent of x_t. Thus, it does not need to be calculated if the distribution is numerically normed. Hence,

$$p(x_t|y_{1:t}) \propto p(y_t|x_t)p(x_t|y_{1:t-1}) \approx \sum_{i=1}^{N} q_{t|t-1}^{(i)} p(y_t|x_t)\delta(x_t - x_t^{(i)}) \tag{3.37}$$

where $\propto$ means proportional to. Here, we can see that the measurement update corresponds to updating the particle weights. This can be done at the same time as normalizing the distribution as

$$q_{t|t}^{(i)} = \frac{q_{t|t-1}^{(i)} p(y_t|x_t^{(i)})}{\sum_{j=1}^{N} q_{t|t-1}^{(j)} p(y_t|x_t^{(j)})}. \tag{3.38}$$

Now the resulting particle distribution is

$$\hat{p}(x_t|y_{1:t}) = q_{t|t}^{(i)} \delta(x_t - x_t^{(i)}). \tag{3.39}$$

Hence, the recursion is complete and starting over with the time update would lead to the MC integration methods. However, this would lead to numerical difficulties which are solved by the resampling step.

Resampling

The MC integration methods will degenerate after a few iterations since all but one particle weight will become close to zero. It can be shown (Doucet, 1998) that this degeneracy is impossible to avoid using these methods. When the PF was derived in Gordon et al. (1993), a resampling step was therefore included to solve this problem. The problem is that the particles can be spread over areas with very low probability and therefore get very low weight. The idea with resampling is to copy likely particles and discard unlikely ones in order to increase the particle density in areas with high probability.

There are several strategies for resampling. The most common are *systematic resampling* (Kitagawa, 1996), *residual resampling* (Liu and Chen, 1998) and *multinomial resampling*.

These steps can now be summarized in Algorithm 3.3.

3.6.2 Discussion

In the derivation of the particle filter above, the marginal distribution $p(x_t|y_{1:t})$ was derived as in the original paper (Gordon et al., 1993). However, it is common in the literature to derive the distribution for the trajectory $p(x_{1:t}|y_{1:t})$, see for instance Doucet et al. (2001). Also in this case it is only the variable x_t that is sampled at time t while the set $\{x_{1:t-1}^{(i)}\}$ remains fixed. During the resampling step the most likely trajectories $x_{1:t}$ are copied and unlikely ones are discarded. This leads to *sample impoverishment* for the old states, and already after a few iterations all particles will be equal. Thus, the estimate for the trajectory cannot be trusted.

The algorithms for estimating the marginal distribution or the trajectory are very similar. The only difference is that there is no need to save the old particles $\{x_{1:t-1}^{(i)}\}$ when estimating the marginal distribution. However, when estimating the marginal distribution, a continuous distribution is available for the prediction during the time update as

$$\hat{p}(x_t|y_{1:t-1}) = \sum_{i=1}^{N} q_{t-1|t-1}^{(i)} p(x_t|x_{t-1}^{(i)}). \tag{3.43}$$

Algorithm 3.3 Particle Filter

1. Initialize the particles according to $x_0^{(i)} \sim p(x_0)$ and set appropriate weights $q_{0|0}^{(i)}$ for all $i = 1, \dots, N$.

2. Time update: Generate new particles according to the proposal distribution

$$x_t^{(i)} \sim \pi(x_t | x_{t-1}^{(i)}, y_t), \quad i = 1, \dots, N. \tag{3.40}$$

   Update the weights according to

$$q_{t|t-1}^{(i)} = \frac{p(x_t^{(i)} | x_{t-1}^{(i)})}{\pi(x_t^{(i)} | x_{t-1}^{(i)}, y_t)} q_{t-1|t-1}^{(i)}, \quad i = 1, \dots, N, \tag{3.41}$$

   which in case $\pi(x_t | x_{t-1}, y_t) = p(x_t | x_{t-1}^{(i)})$ simplifies to $q_{t|t-1}^{(i)} = q_{t-1|t-1}^{(i)}$.

3. Measurement update: Calculate importance weights according to

$$q_{t|t}^{(i)} = \frac{q_{t|t-1}^{(i)} p(y_t | x_t^{(i)})}{\sum_{j=1}^{N} q_{t|t-1}^{(j)} p(y_t | x_t^{(j)})}, \quad i = 1, \dots, N. \tag{3.42}$$

4. Resampling: Use a method of choice, e.g., the ones discussed in Section 3.6.1.

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

This allows us to sample from a proposal distribution of the form

$$x_t^{(i)} \sim \sum_{i=1}^{N} q_{t-1|t-1}^{(i)} \pi(x_t | x_{t-1}^{(i)}, y_t), \tag{3.44}$$

where the weights are updated according to

$$q_{t|t-1}^{(i)} = \frac{\sum_{j=1}^{N} q_{t-1|t-1}^{(j)} p(x_t^{(i)} | x_{t-1}^{(j)})}{N \sum_{j=1}^{N} q_{t-1|t-1}^{(j)} \pi(x_t^{(i)} | x_{t-1}^{(j)}, y_t)}. \tag{3.45}$$

The resulting filter is called the *Marginal Particle Filter* and was derived in Klaas et al. (2005). For this filter, the resampling step is not necessary since the same type of operation is done directly in the sampling step. The numerical properties are also shown to be better. The disadvantage is that the complexity of the sampling step is $\mathcal{O}(N^2)$ compared to $\mathcal{O}(N)$ for the ordinary PF.

3.7 Rao-Blackwellized Particle Filter

The particle filter requires large amounts of particles as the state dimension increases to maintain a good approximation of the filtering distribution. It is therefore desirable to keep the state dimension treated by the particle filter as low as possible. The Rao-Blackwellized Particle Filter (RBPF) reduces the dimension of the state space treated by the particle filter by partitioning the state space into two parts as $x_t = ({x_t^p}^T \; {x_t^k}^T)^T$. The posterior distribution can then also be partitioned into two parts as

$$p(x_t^p, x_t^k | y_{1:t}) = p(x_t^p | y_{1:t}) p(x_t^k | x_t^p, y_{1:t}). \tag{3.46}$$

The idea is to use different filters for computing the two parts of the posterior. If the system is linear given the states x_t^p (i.e., conditionally linear), then the states x_t^k can be calculated using a KF. The dimension in the particle filter is then limited to the dimension of x_t^p. The RBPF is treated in Chen and Liu (2000), Andrieu and Doucet (2002) and is also sometimes referred to as the *Marginalized Particle Filter* (Schön, 2006, Schön et al., 2005). The implementation of the RBPF in this section can be seen as the filter bank formulation derived in Hendeby (2008) and Hendeby et al. (2007).

3.7.1 Derivations

The RBPF traditionally estimates the distribution for the full trajectory of the nonlinear states, i.e., $p(x_{1:t}^p, x_t^k | y_{1:t})$. The derivation in this thesis is done for the marginal distribution $p(x_t^p, x_t^k | y_{1:t})$. The motivation is the same as for the regular PF discussed in Section 3.6.2.

The following model is used for RBPF since it fulfills the requirement on being conditionally linear

$$x_t^p = f_{t-1}^p(x_{t-1}^p) + F_{t-1}^p(x_{t-1}^p) x_{t-1}^k + G_{t-1}^p(x_{t-1}^p) w_{t-1}^p, \tag{3.47a}$$

$$x_t^k = f_{t-1}^k(x_{t-1}^p) + F_{t-1}^k(x_{t-1}^p) x_{t-1}^k + G_{t-1}^k(x_{t-1}^p) w_{t-1}^k, \tag{3.47b}$$

$$y_t = h_t(x_t^p) + H_t(x_t^p) x_t^k + e_t, \tag{3.47c}$$

where the noises are assumed to be zero-mean Gaussian with the following covariance

$$\begin{pmatrix} w_t^p \\ w_t^k \end{pmatrix} \sim \mathsf{N}(0, \underbrace{\text{diag}(Q_t^p, Q_t^k)}_{=Q_t}), \quad e_t \sim \mathsf{N}(0, R_t). \tag{3.48}$$

In order to simplify the notation, we will make use of the following notation

$$f_t(x_t^p) = \begin{pmatrix} f_t^p(x_t^p) \\ f_t^k(x_t^p) \end{pmatrix}, \ F_t = \begin{pmatrix} F_t^p(x_t^p) \\ F_t^k(x_t^p) \end{pmatrix}, \ G_t = \begin{pmatrix} G_t^p(x_t^p) & 0 \\ 0 & G_t^k(x_t^p) \end{pmatrix}, \ w_t = \begin{pmatrix} w_t^p \\ w_t^k \end{pmatrix}.$$

The dynamic model in (3.47a–3.47b) can then be written as

$$x_t = \begin{pmatrix} x_t^p \\ x_t^k \end{pmatrix} = f_{t-1}(x_{t-1}^p) + F_{t-1} x_{t-1}^k + G_t w_{t-1}. \tag{3.49}$$

Initialize

Start with the initial distribution

$$\hat{p}(x_{t-1}^p, x_{t-1}^k | y_{1:t-1}) = \sum_{i=1}^{N} q_{t-1}^{(i)} \, \mathsf{N}(x_{t-1}^k; x_{t-1|t-1}^{k(i)}, P_{t-1|t-1}^{(i)}) \delta(x_{t-1}^p - x_{t-1}^{p(i)}). \tag{3.50}$$

Time Update

The time update starts with computing the one-step ahead prediction by using (3.14) as

$$p(x_t^p, x_t^k | y_{1:t-1}) = \iint p(x_t^p, x_t^k | x_{t-1}^p, x_{t-1}^k) \underbrace{p(x_{t-1}^p, x_{t-1}^k | y_{1:t-1})}_{(3.50)} \, dx_{t-1}^k \, dx_{t-1}^p. \tag{3.51}$$

Since $p(x_{t-1}^p, x_{t-1}^k | y_{1:t-1})$ is a Gaussian sum but sampled in the dimensions of x^p, this integral can be evaluated analytically. The obtained solution is a sum of Gaussians and the evaluation is similar to the way a KF is derived. The result is

$$\hat{p}(x_t^p, x_t^k | y_{1:t-1}) = \sum_i q_{t-1}^{(i)} \, \mathsf{N}(x_t; \bar{x}_{t|t-1}^{(i)}, \bar{P}_{t|t-1}^{(i)}), \tag{3.52}$$

where

$$\bar{x}_{t|t-1}^{(i)} = f_{t-1}(x_{t-1|t-1}^{p(i)}) + F_{t-1}^{(i)} x_{t-1|t-1}^{k(i)}, \tag{3.53a}$$

$$\bar{P}_{t|t-1}^{(i)} = F_{t-1}^{(i)} P_{t-1|t-1}^{(i)} F_{t-1}^{(i)T} + G_{t-1}^{(i)} Q_{t-1} G_{t-1}^{(i)T}, \tag{3.53b}$$

and the following split can be made analogously to the split of x_t,

$$\bar{x} = \begin{pmatrix} \bar{x}^p \\ \bar{x}^k \end{pmatrix}, \quad \bar{P} = \begin{pmatrix} \bar{P}^p & \bar{P}^{pk} \\ \bar{P}^{pkT} & \bar{P}^k \end{pmatrix}.$$

This Gaussian mixture cannot be handled in the measurement update since the distribution is continuous in the nonlinear part x^p. Therefore, the Gaussian distributions in (3.52) are split into two parts

$$\hat{p}(x_t^p, x_t^k | y_{1:t-1}) = \sum_{i=1}^{N} q_{t-1}^{(i)} \, \mathsf{N}(x_t^{k(i)}; x_{t|t-1}^{k(i)}, P_{t|t-1}^{(i)}) \, \mathsf{N}(x_t^p; \bar{x}_{t|t-1}^{p(i)}, \bar{P}_{t|t-1}^{p(i)}), \tag{3.54}$$

where

$$x_{t|t-1}^{k(i)} = \bar{x}_{t|t-1}^{k(i)} - \bar{P}_{t|t-1}^{pk(i)T} (\bar{P}_{t|t-1}^{p(i)})^{-1} (x_t^p - \bar{x}_{t|t-1}^{p(i)}), \tag{3.55a}$$

$$P_{t|t-1}^{(i)} = \bar{P}_{t|t-1}^{k(i)} - \bar{P}_{t|t-1}^{pk(i)T} (\bar{P}_{t|t-1}^{p(i)})^{-1} \bar{P}_{t|t-1}^{pk(i)}. \tag{3.55b}$$

Note that the first part is conditioned on the second and the calculations are just a matter of completing the squares in the exponential of the Gaussian distributions. For each term in the sum (3.54) the distribution for x_t^k is sampled with one sample

$$x_t^{p(i)} \sim \mathsf{N}(\bar{x}_{t|t-1}^{p(i)}, \bar{P}_{t|t-1}^{p(i)}). \tag{3.56}$$

Thus the distribution can finally be approximated as

$$\hat{p}(x_t^p, x_t^k | y_{1:t-1}) = \sum_{i=1}^{N} q_{t-1}^{(i)} \, \mathsf{N}(x_t^k; x_{t|t-1}^{k(i)}, P_{t|t-1}^{(i)}) \delta(x_t^p - x_t^{p(i)}), \tag{3.57}$$

where $x_{t|t-1}^{k(i)}$ and $P_{t|t-1}^{(i)}$ are calculated using (3.55) for a specific particle $x_t^p = x_t^{p(i)}$.

Measurement update

Split the measurement updated distribution as follows

$$p(x_t^p, x_t^k | y_{1:t}) = p(x_t^p | y_{1:t}) p(x_t^k | x_t^p, y_{1:t}). \tag{3.58}$$

Now we can compute the two parts separately, start with the KF part which is calculated for each particle

$$p(x_t^k | x_t^{p(i)}, y_{1:t}) = \frac{p(y_t | x_t^{p(i)}, x_t^k) p(x_t^k | x_t^{p(i)}, y_{1:t-1})}{p(y_t | y_{1:t-1})}. \tag{3.59}$$

All involved densities here are Gaussian, so this can be identified as a KF measurement update resulting in the following solution

$$p(x_t^k | x_t^{p(i)}, y_{1:t}) = \mathsf{N}(x_t^k; x_{t|t}^{k(i)}, P_{t|t}^{(i)}) \tag{3.60}$$

where

$$x_{t|t}^{k(i)} = x_{t|t-1}^{k(i)} + K_t^{(i)} (y_t - \hat{y}_t^{(i)}), \tag{3.61}$$

$$P_{t|t}^{(i)} = P_{t|t-1}^{(i)} - K_t^{(i)} S_t^{(i)} K_t^{(i)T}, \tag{3.62}$$

and

$$\hat{y}_t^{(i)} = h_t(x_t^{p(i)}) + H_t(x_t^{p(i)})x_{t|t-1}^{k(i)}, \tag{3.63}$$

$$S_t^{(i)} = H_t(x_t^{p(i)})P_{t|t-1}H_t(x_t^{p(i)})^T + R_t, \tag{3.64}$$

$$K_t^{(i)} = P_{t|t-1}H_t(x_t^{p(i)})^T(S_t^{(i)})^{-1}. \tag{3.65}$$

The measurement update for the nonlinear states corresponds to calculate

$$p(x_t^p|y_{1:t}) = \frac{p(y_t|x_t^p)p(x_t^p|y_{1:t-1})}{p(y_t|y_{1:t-1})}, \tag{3.66}$$

where it is observed that the denominator is only a normalizing constant since it is independent of x_t^p. Thus, it does not need to be calculated if the distribution is numerically normed. Again, the calculations are done per particle

$$\begin{aligned} p(x_t^p|y_{1:t}) \propto p(y_t|x_t^p)p(x_t^p|y_{1:t-1}) &= \int p(y_t|x_t^p, x_t^k)p(x_t^k|x_t^p, y_{1:t})\, dx_t^k p(x_t^p|y_{1:t-1}) \\ &= \int p(y_t|x_t^p, x_t^k)\, \mathsf{N}(x_t^k; x_{t|t}^{k(i)}, P_{t|t}^{(i)})\, dx_t^k q_{t-1}^{(i)}\delta(x_t^p - x_t^{p(i)}) \\ &= q_{t-1}^{(i)}\, \mathsf{N}(y_t; \hat{y}_t^{(i)}, S_t^{(i)})\delta(x_t^p - x_t^{p(i)}), \end{aligned} \tag{3.67}$$

where $\propto$ means proportional to. Here, we can see that the measurement update corresponds to updating the particle weights. This can be done at the same time as normalizing the distribution as

$$q_t^{(i)} = \frac{\mathsf{N}(y_t; \hat{y}_t^{(i)}, S_t^{(i)})q_{t-1}^{(i)}}{\sum_{j=1}^N \mathsf{N}(y_t; \hat{y}_t^{(j)}, S_t^{(j)})q_{t-1}^{(j)}}. \tag{3.68}$$

The final distribution is then given by

$$\hat{p}(x_t^p, x_t^k|y_{1:t}) = \sum_{i=1}^N q_t^{(i)}\, \mathsf{N}(x_t^k; x_{t|t}^{k(i)}, P_{t|t}^{(i)})\delta(x_t^p - x_t^{p(i)}). \tag{3.69}$$

Resampling

The resampling step is analogous to the one for the ordinary PF. See Section 3.6.1 for more information.

3.8 Hypothesis Testing

A common problem is to detect whether a change of some type has occurred. Common applications are in the area of disturbance detection or fault detection. In this dissertation, the Papers B and C address this issue. Paper B detects magnetic disturbances on a magnetometer and Paper C detects errors in extraction of features in an image.

The detection problem is often formulated in a way that it is a question of deciding between a number of hypotheses. To make the decision, measurement data from some

---

Algorithm 3.4 Rao-Blackwellized Particle Filter (RBPF)

---

1. Initialize: For all particles $i = 1, \dots, N$, let $x_{0|0}^{k(i)} = x_0^k$, $P_{0|0}^{(i)} = P_0$ and $x_0^{p(i)} \sim p(x_0^p)$.

2. Time update: Compute the Gaussian mixture by

   $$\bar{x}_{t|t-1}^{(i)} = f_{t-1}(x_{t-1}^{p(i)}) + F_{t-1}^{(i)} x_{t-1|t-1}^{k(i)}, \tag{3.70a}$$

   $$\bar{P}_{t|t-1}^{(i)} = F_{t-1}^{(i)} P_{t-1|t-1}^{(i)} F_{t-1}^{(i)T} + G_{t-1}^{(i)} Q_{t-1} G_{t-1}^{(i)T}, \tag{3.70b}$$

   where the following split can be made $\bar{x} = \begin{pmatrix} \bar{x}^p \\ \bar{x}^k \end{pmatrix}$ and $\bar{P} = \begin{pmatrix} \bar{P}^p & \bar{P}^{pk} \\ \bar{P}^{pkT} & \bar{P}^k \end{pmatrix}$. Sample the PF states according to

   $$x_t^{p(i)} \sim \mathsf{N}(\bar{x}_{t|t-1}^{p(i)}, \bar{P}_{t|t-1}^{p(i)}). \tag{3.70c}$$

   Update the KF states according to

   $$x_{t|t-1}^{k(i)} = \bar{x}_{t|t-1}^{k(i)} - \bar{P}_{t|t-1}^{pk(i)T} (\bar{P}_{t|t-1}^{p(i)})^{-1} (x_t^{p(i)} - \bar{x}_{t|t-1}^{p(i)}), \tag{3.70d}$$

   $$P_{t|t-1}^{(i)} = \bar{P}_{t|t-1}^{k(i)} - \bar{P}_{t|t-1}^{pk(i)T} (\bar{P}_{t|t-1}^{p(i)})^{-1} \bar{P}_{t|t-1}^{pk(i)}. \tag{3.70e}$$

3. Measurement update:

   $$q_t^{(i)} = \frac{\mathsf{N}(y_t; \hat{y}_t^{(i)}, S_t^{(i)}) q_{t-1}^{(i)}}{\sum_{j=1}^{N} \mathsf{N}(y_t; \hat{y}_t^{(j)}, S_t^{(j)}) q_{t-1}^{(j)}} \tag{3.71a}$$

   $$x_{t|t}^{k(i)} = x_{t|t-1}^{k(i)} + K_t^{(i)} (y_t - \hat{y}_t^{(i)}) \tag{3.71b}$$

   $$P_{t|t}^{(i)} = P_{t|t-1}^{(i)} - K_t^{(i)} S_t^{(i)} K_t^{(i)T} \tag{3.71c}$$

   where
   $\hat{y}_t^{(i)} = h_t(x_t^{p(i)}) + H_t(x_t^{p(i)}) x_{t|t-1}^{k(i)}$, $S_t^{(i)} = H_t(x_t^{p(i)}) P_{t|t-1} H_t(x_t^{p(i)})^T + R_t$ and $K_t^{(i)} = P_{t|t-1} H_t(x_t^{p(i)})^T (S_t^{(i)})^{-1}$.

4. Resampling: Use a method of choice, e.g., the ones discussed in Section 3.6.1, update $q_t^{(i)}$.

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

---

sensors and a model of the process are often provided. If there are two hypotheses to decide between, the problem is known as a *binary hypothesis test*. To detect faults in a system, the hypotheses could be formulated as

$$\begin{aligned} \mathcal{H}_0 : &\quad \text{The system works as normal,} \\ \mathcal{H}_1 : &\quad \text{The system is faulty,} \end{aligned}$$

where the hypotheses $\mathcal{H}_0$ and $\mathcal{H}_1$ usually are referred to as *null hypothesis* and *alternative hypothesis* respectively. Using sensor measurements and the model of the system a *test statistic* can be derived. The test statistic will have different distributions in case a fault is present or not. If the distributions are completely known in both cases, we will have a *simple hypothesis test*. Otherwise, if some parameters are unknown, we have a *composite hypothesis test*.

3.8.1 Simple Hypothesis Testing

Consider the case where the binary variable $\theta = \{\theta_0, \theta_1\}$ is measured with additive noise e which has a known distribution,

$$y = \theta + e.$$

The hypothesis test

$$\begin{aligned} \mathcal{H}_0 : &\quad \theta = \theta_0, \\ \mathcal{H}_1 : &\quad \theta = \theta_1, \end{aligned}$$

is an example of a simple hypothesis test if the parameters θ_0 and θ_1 are known. The probability of the measurement can then be calculated depending on which hypothesis that is assumed true. One approach to selection between the hypotheses is to select the hypothesis that gives the highest probability for the measurement. Hence, if $p(y|\,\mathcal{H}_1) > p(y|\,\mathcal{H}_0)$, the alternative hypothesis is decided. Another approach would be to be more restrictive with causing an alarm, so the requirement for an alarm is that $p(y|\,\mathcal{H}_1) > \gamma p(y|\,\mathcal{H}_0)$ where $\gamma > 1$. This can also be expressed as a ratio between the probabilities, the *likelihood ratio*

$$\frac{p(y|\,\mathcal{H}_1)}{p(y|\,\mathcal{H}_0)} > \gamma.$$

In order to statistically describe these approaches, some definitions have to be made.

Definition 3.1. Define the following detection related terms

- False alarm (FA) is to decide $\mathcal{H}_1$ when $\mathcal{H}_0$ is true.
- Miss (M) is to decide $\mathcal{H}_0$ when $\mathcal{H}_1$ is true.
- Probability of false alarm (P_{FA}); $P_{FA} = P(\mathcal{H}_1 \,|\, \mathcal{H}_0)$
- Probability of miss (P_M); $P_M = P(\mathcal{H}_0 \,|\, \mathcal{H}_1)$
- Probability of detection (P_D); $P_D = P(\mathcal{H}_1 \,|\, \mathcal{H}_1) = 1 - P_M$

When designing a test to decide between the hypotheses, a trade-off between high P_D and low P_{FA} has to be made. If it is decided to have P_{FA} on a certain level, what is the optimal test? This question is answered by the Neyman-Pearson, NP, lemma, which was first published in the articles Neyman and Pearson (1928a,b). Here, the lemma is presented in Theorem 3.1. The NP-lemma is referred to as the *Likelihood Ratio Test*, LRT, and is the most powerful test for simple hypotheses. Most powerful means that it maximizes P_D, given a P_{FA}.

Theorem 3.1 (Neyman-Pearson lemma)
To maximize P_D for a given $P_{FA} = \alpha$ decide $\mathcal{H}_1$ if

$$L(y) = \frac{p(y|\mathcal{H}_1)}{p(y|\mathcal{H}_0)} > \gamma, \tag{3.72}$$

where the threshold γ is given by

$$P_{FA} = \int\limits_{y:L(y)>\gamma} p(y|\mathcal{H}_0)\,dy = \alpha. \tag{3.73}$$

Proof: See (Kay, 1998, Appendix 3A). ☐

3.8.2 Composite Hypothesis Testing

Consider the measurement

$$y = \theta + e, \tag{3.74}$$

where the measurement noise e has a known distribution but θ is unknown. The one-sided hypothesis

$$\mathcal{H}_0 : \quad \theta = 0, \tag{3.75}$$

$$\mathcal{H}_1 : \quad \theta > 0, \tag{3.76}$$

will have an unknown parameter in the distribution of y under the alternative hypothesis. This test is therefore called a composite hypothesis test. It is sometimes possible to find an optimal detector even for composite hypotheses. A detector that yields the highest P_D given P_{FA} for all values of θ is known as *Uniformly Most Powerful*, UMP. A two-sided hypothesis test could be

$$\mathcal{H}_0 : \quad \theta = 0, \tag{3.77}$$

$$\mathcal{H}_1 : \quad \theta \neq 0, \tag{3.78}$$

where the alternative hypothesis has two test regions for the parameter θ. It can be shown that a UMP test does not exist for a two-sided hypothesis test, see Kay (1998). Thus, in those cases suboptimal detectors have to be implemented.

To construct a detector for a composite hypothesis test, there are two major approaches:

1. Marginalization or Bayesian approach. If the unknown parameter has a known distribution it can be marginalized.
2. Generalized Likelihood Ratio Test, GLRT.

The approaches will be discussed in the following sections.

Marginalization

Consider again the measurement

$$y = \theta + e, \tag{3.79}$$

where e is noise with known distribution but θ is unknown. The hypotheses are

$$\mathcal{H}_0 : \quad \theta = \theta_0, \tag{3.80a}$$

$$\mathcal{H}_1 : \quad \theta = \theta_1, \tag{3.80b}$$

where θ_0 and θ_1 are unknown. Now, there are unknown parameters in the distribution of y for both hypotheses. The marginal density function of y can be computed as

$$p(y|\mathcal{H}_0) = \int_{-\infty}^{\infty} p(y|\mathcal{H}_0, \theta_0) p(\theta_0)\, d\theta_0, \tag{3.81a}$$

$$p(y|\mathcal{H}_1) = \int_{-\infty}^{\infty} p(y|\mathcal{H}_1, \theta_1) p(\theta_1)\, d\theta_1. \tag{3.81b}$$

This can be thought of as computing the “expected distribution, $p(y)$”. Inserting this density function in the LR test gives

$$\tilde{L}(y) = \frac{p(y|\mathcal{H}_1)}{p(y|\mathcal{H}_0)}. \tag{3.82}$$

Generalized Likelihood Ratio

Another way of eliminating the unknown parameters in (3.80) is to use the *maximum likelihood estimate* of the parameter. The parameters can then be calculated as

$$\hat{\theta}_i = \arg\max_{\theta_i} p(y|\theta_i). \tag{3.83}$$

The estimated parameters $\hat{\theta}_i$, where $i \in \{0, 1\}$, are then inserted into the Likelihood Ratio test provided in the NP-lemma which yields the Generalized Likelihood Ratio test,

$$\hat{L}(y) = \frac{p(y|\hat{\theta}_1)}{p(y|\hat{\theta}_0)}. \tag{3.84}$$

Even though the NP-test is optimal, the GLRT is not always optimal but often works well in practice. The test is shown to be optimal when infinite information is available, but this is only of theoretical interest (Lehmann, 1986). An example of a GLRT is given in Example 3.1.

Example 3.1: Detection with GLR

Paper B deals with detection of magnetic disturbances on a magnetometer. A residual which can be used for detection is constructed on the form

$$r = W^T \mathbb{Z} \sim \mathsf{N}(W^T\theta, I), \tag{3.85}$$

where r is a vector of dimension n_r, W^T is a projector, $\mathbb{Z}$ contains a batch of measurements, and θ is the fault parameter. The test to see if there is a fault present or not, can be formulated as the hypothesis test

$$\mathcal{H}_0 : \quad \theta = 0, \tag{3.86a}$$

$$\mathcal{H}_1 : \quad \theta \neq 0. \tag{3.86b}$$

This is a composite hypothesis test since there is an unknown parameter θ. The GLRT can be used for detection and the ML-estimate of θ is

$$\begin{aligned}\hat{\theta} = \arg\max_\theta p(r|\theta) = \arg\max_\theta \frac{1}{(2\pi)^{n_r/2}} e^{-\frac{1}{2}(r-W^T\theta)^T(r-W^T\theta)} \\ = \arg\max_\theta \|r - W^T\theta\|_2^2 = (W^T)^\dagger r \end{aligned} \tag{3.87}$$

where $(W^T)^\dagger$ denotes the Moore-Penrose pseudo inverse of W^T (see Golub and van Loan (1996)). The logarithm of the GLR test statistic in (3.84) will also serve as a test statistic since $\log$ is a strictly increasing function. Thus, the test statistic is

$$\begin{aligned}\hat{L}(r) &= 2\log\frac{p(r|\theta=\hat{\theta})}{p(r|\theta=0)} \\ &= 2\log\frac{e^{-\frac{1}{2}(r-W^T(W^T)^\dagger r)^T(r-W^T(W^T)^\dagger r)}}{e^{-\frac{1}{2}r^Tr}} = r^T\mathcal{P}_{W^T}r, \end{aligned} \tag{3.88}$$

where $\mathcal{P}_{W^T} = W^T(W^T)^\dagger$ is an orthogonal projection (Meyer, 2000). Since the eigenvalues of a projection matrix is 1, the test statistic is distributed as the non-central chi-square distribution

$$\hat{L}(r) = \sim \chi'^2_\nu(\lambda), \tag{3.89}$$

where $\nu = \operatorname{rank}(W^T)$ and

$$\lambda = (W^T\theta)^T\mathcal{P}_{W^T}W^T\theta = (W^T\theta)^TW^T\theta. \tag{3.90}$$

Observe that $\lambda = 0$ in the fault-free case and then the test statistic is distributed according to the central chi-square distribution $L \sim \chi^2_\nu$. The threshold is then chosen from the chi-square distribution (see Appendix A.2) so that the fault-free hypothesis is rejected erroneously only with a small probability.

4

Simultaneous Localization and Mapping

A CHALLENGING PROBLEM for the robotics community over the last 20 years has been the Simultaneous Localization and Mapping (SLAM) problem. The problem is to have a robot can localize itself and at the same time build a map of its environment. The solution to this problem would be an enabler for true autonomous systems that can explore previously unknown environments. Although satellite systems such as GPS, Galileo and GLONASS can be used for positioning, it is often an advantage and sometimes necessary to have a positioning system that does not require any external infrastructure. Such a system does not need a predefined map of its surroundings, it will work where the satellite signal is jammed or shadowed, and it will also work at remote places like other planets.

This chapter is divided into three sections. The first poses the SLAM problem, analyzes the difficulties and points to solutions in the literature. The following two sections comment more closely on two of the most common SLAM algorithms, EKF-SLAM and FastSLAM.

4.1 Problem Formulation

A general formulation of the SLAM problem is to have a robot to localize itself and at the same time build a map of its environment. The map could be represented in different ways, but this chapter will primarily treat *feature* or *landmark* based maps. It is popular to treat the SLAM problem in the Bayesian framework, which will be adopted here as well. For a feature based SLAM problem, the Bayesian estimation problem can be mathematically defined as at time t finding the distribution

$$p(x_t^r, m|y_{1:t}), \tag{4.1}$$

where x_t^r is the state of the robot, m is the map state and $y_{1:t}$ are the measurements up to time t. A general state-space model for the SLAM-problem with static map is nonlinear and can be written as

$$x_t^r = f^r(x_{t-1}^r, u_{t-1}, v_{t-1}), \tag{4.2a}$$
$$m_t = m_{t-1}, \tag{4.2b}$$
$$y_t^i = h(x_t^r, m_t^i) + e_t, \tag{4.2c}$$

where u_t is the input signal, v_t is the process noise, and e_t is the measurement noise.

The SLAM problem has been extensively studied since the mid 1980's by many talented researchers. Many solutions have been proposed during the years, but there are still many open questions to be answered before SLAM can be used robustly in unstructured environments. Three major difficulties with SLAM are:

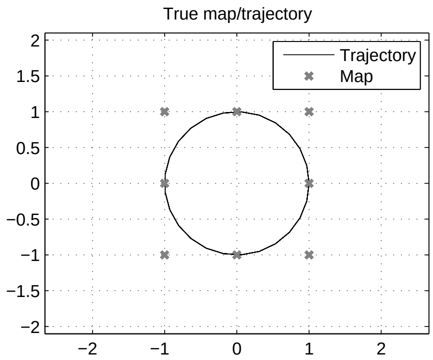
1. Dependencies and high dimensionality
2. Nonlinearities
3. Processing of data from sensors that sense the environment

The first point is motivated with an example. If the system in (4.2) is linear, the solution to the filtering problem is given by the Kalman Filter (KF). An example of a model for a linear SLAM problem is given by

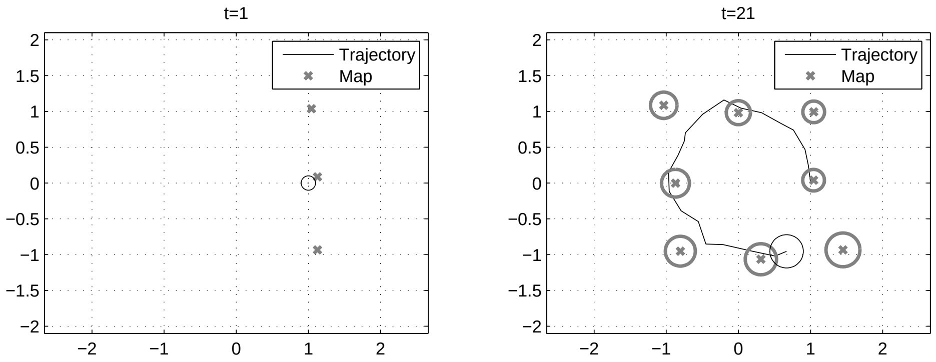
$$x_t^r = \begin{pmatrix} \mathsf{p}_t \\ \mathsf{v}_t \end{pmatrix} = \begin{pmatrix} I & TI \\ 0 & I \end{pmatrix} \begin{pmatrix} \mathsf{p}_{t-1} \\ \mathsf{v}_{t-1} \end{pmatrix} + \begin{pmatrix} \frac{T^2}{2} I \\ TI \end{pmatrix} v_{t-1}, \tag{4.3a}$$
$$m_t^i = m_{t-1}^i, \tag{4.3b}$$
$$y_t^i = \mathsf{p}_t - m_t^i + e_t, \quad \text{if } \|\mathsf{p}_t - m_t^i\| \leq 1, \tag{4.3c}$$

where p_t is the robot position, v_t the robot velocity and m_t^i the position of feature i. It is assumed that the relative position to the features can be measured if the feature is closer than 1 length unit, which is motivated since sensors often have limited range. Important to note about the KF solution to this problem is that the features are strongly correlated with each other and the robot states. This is intuitive since the position of the features are measured relative to the robot position, thus the position of the features will depend on the robot trajectory and vice versa. This dependence is also what makes SLAM work. Figure 4.1 shows the event of a *loop closure*, i.e., the robot returns to a previously visited area. Before the loop closure (Figure 4.1c), the feature positions have drifted away due to accumulation of errors when only new features are seen. As the initial feature is seen (Figure 4.1d), the entire map is corrected thanks to the cross-correlations present in the map. The filter converges to a solution where the relative map has zero uncertainty, only the initial uncertainty of the robot position remains. Discussions about the properties of KF-SLAM and convergence can be found in Dissanyake et al. (2001) and Csorba (1997). The fact that the map is highly correlated could be seen in this example. This important property is also what make the problem hard to decompose. As the dimension of the map m grows, the covariance matrix becomes large, which leads to computational problems.

The SLAM problem is seldom linear, both the robot dynamics and the measurement relations are nonlinear in most real world examples. To solve the nonlinear filtering problem, approximations are used. The earliest attempts of solving the SLAM problem were

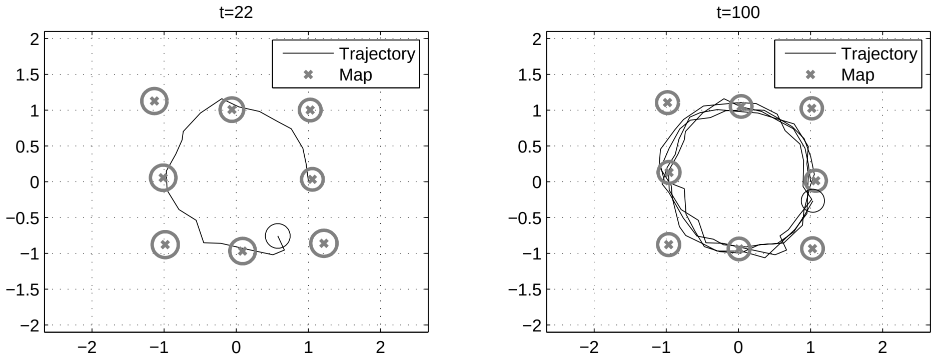


(a) *True map and true robot trajectory.*



(b) *Initial map*

(c) *Just before loop closing*



(d) *Just after loop closing*

(e) *After four laps*

Figure 4.1: *Example of a solution to the linear* SLAM *problem. The robot trajectory and the current covariance are plotted together with the mapped features (crosses) and their associated covariance (ellipses).*

done using an Extended Kalman Filter (EKF) and theory presented in Smith et al. (1990) and Durrant-Whyte (1988), see also Section 4.2. Another approach is built on the Particle Filter (PF) where the true solution to the SLAM problem could be obtained if infinitely many particles were used, see Montemerlo et al. (2002). In reality this is not possible due to computational constraints, so a finite number of particles are used which gives an approximative solution. Besides handling nonlinearities, the PF solution has the advantage of relatively low complexity with respect to the number of features. The PF solution is referred to as FastSLAM and is further discussed in Section 4.3.

It is of course necessary to have a sensor that gives information about the surrounding environment to solve the SLAM problem. Such a sensor could be a camera, radar, sonar, laser etc. The problem with these sensors is that they give a very high dimensional measurement that is hard to use unprocessed in a filter. Take the camera sensor as an example, the sensor measurement consists of several thousands of pixels that cannot be used directly in the filter. Instead the image is preprocessed, interest points or features that easily can be found in consecutive images are identified. Their positions in the image are then used in the filter. How to select features that can be found again when the robot returns to the same location from another angle or in different light settings is a very challenging problem.

For the SLAM problem, there are different approaches regarding which sensors to include. Some only uses one sensor, such as a camera, see e.g., Davison et al. (2007), Davison (2003), Davison et al. (2004), Eade and Drummond (2006). Another approach is to combine several sensors, such as an Inertial Measurement Unit (IMU) with a camera and a barometer, this approach is taken in Paper A. Papers C and E also consider the use of a camera in combination with other sensors. The advantage with combining the vision sensor with other sensors is that there is less need for overlap between the images. Thus, the frame rate of the camera can be lower and computer intensive image processing is saved. Furthermore, the IMU allows for the ability to handle rapid movements.

A good overview of different SLAM approaches and the history of SLAM is given in the paper series Durrant-Whyte and Bailey (2006) and Bailey and Durrant-Whyte (2006). The recent book (Thrun et al., 2005) discusses many topics in robotics and SLAM. In the subsequent sections, two of the most common solutions to the SLAM problem will be discussed.

4.2 EKF-SLAM

The earliest approaches to solving the SLAM problem was by applying an EKF to the model in (4.2). This solution has proved to work well when the number of features is small. The convergence and consistency properties of the EKF based approach has been analyzed in, for instance, Huang and Dissanayake (2007) and Bailey et al. (2006a). Consistency can only be guaranteed in the linear case and the estimates are often overly confident.

The EKF based approach has proved to be very sensitive to incorrect association of the features. It is therefore important that incorrect associations are detected, see Neira and Tardos (2001) and Paper C.

The quadratic complexity (see Section 4.2.1) of the EKF-SLAM algorithm has made it computationally demanding for big maps. Several variants of the EKF-SLAM approach have therefore been proposed. Using the information form of the EKF, sparseness of the information matrices can be used to lower the complexity (Thrun et al., 2004). Another approach is to divide the map into regional maps in a hierarchical structure and thus treat smaller subproblems, see for example Jensfelt (2001). Other authors have noticed that it is possible to remove a large number of features and still maintain consistency (Dissanayake et al., 2002).

4.2.1 Algorithm

The EKF-SLAM algorithm is given in Algorithm 4.1 and assumes a linearized model as for the EKF in Section 3.5. The Jacobians of the model in (4.2) are defined as:

$$F_t^r = \left.\frac{\partial f(x^r,u,v)}{\partial x^r}\right|_{(x^r,u,v)=(\hat{x}^r_{t|t},u_t,0)} \qquad G_t \triangleq \left.\frac{\partial f(x^r,u,v)}{\partial v}\right|_{(x^r,u,v)=(\hat{x}^r_{t|t},u_t,0)} \tag{4.4}$$

$$H_t = \left.\frac{\partial h(x^r,m)}{\partial (x^r,m)}\right|_{(x^r,m)=(\hat{x}^r_{t|t-1},m_{t|t-1})} \tag{4.5}$$

The first step of the algorithm is to detect features and associate them with previously observed features. This step is of course very sensor specific. Once the features have been associated, the previously seen features can be used in the measurement update step. The map is incrementally built as new features are observed. A new feature i is initialized with $P_{m^i} = \infty$. All information about the feature positions are then inferred by the measurements, making sure that the cross-correlations become correct. In practice, it is not possible to set the covariance to infinity. Setting the covariance to a large value and the initial estimate by inverting the measurement equation yields a good approximation. The time update has been done noting that only the covariances affecting the robot states have to be updated.

The complexity of the EKF is normally $\mathcal{O}(n_x^3)$, where n_x is the state dimension. In EKF-SLAM, the special structure can be utilized to obtain a lower complexity. Note that multiplication of an $n_1 \times n_2$ and an $n_2 \times n_3$ matrix has complexity $\mathcal{O}(n_1 n_2 n_3)$. Then it easy to see that the time update step has complexity $\mathcal{O}(n_r^3 + n_r^2 n_m)$, where n_r is the number of states for the robot and n_m is the number of states in the map so $n_x = n_r + n_m$. For the measurement update, the special structure of the measurement can be utilized. The Jacobian of the measurement equation (4.5) will have the following sparse structure

$$H_t = \begin{pmatrix} H_{11} & H_{12} & 0 & \cdots & 0 & 0 & \cdots \\ H_{21} & 0 & 0 & \cdots & H_{22} & 0 & \cdots \\ \vdots & & & & & & \end{pmatrix}. \tag{4.6}$$

The structure arises from the fact that each measured feature is only a function of the robot state and the measured feature state. Using this, the complexity of the measurement update step can be calculated to be $\mathcal{O}(n_y^3 + n_x n_y^2 + n_x^2 n_y)$, where n_y is the measurement dimension. Since $n_x = n_r + n_m$, the computational complexity with respect to the size of the map is $\mathcal{O}(n_m^2)$.

Algorithm 4.1 EKF-SLAM

An initial state consisting of only robot states, $\hat{x}_{0|-1} = x_0$, and an initial covariance, $P_{0|-1} = P_0$, are given. Then the algorithm is given by

1. Detect features and associate
2. Measurement update

$$K_t = P_{t|t-1} H_t^T (H_t P_{t|t-1} H_t^T + R_t)^{-1} \tag{4.7a}$$

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t\big(y_t - h(\hat{x}_{t|t-1})\big) \tag{4.7b}$$

$$P_{t|t} = (I - K_t H_t) P_{t|t-1} \tag{4.7c}$$

3. Add new features

$$\hat{x}_{t|t} := \begin{pmatrix} \hat{x}_{t|t} \\ 0 \end{pmatrix} \tag{4.7d}$$

$$P_{t|t} := \begin{pmatrix} P_{t|t} & 0 \\ 0 & \infty I \end{pmatrix} \tag{4.7e}$$

4. Time update

$$\hat{x}^r_{t+1|t} = f(\hat{x}^r_t, u_t) \tag{4.7f}$$

$$P_{t+1|t} = \begin{pmatrix} F^r_t P^r_{t|t} F^{rT}_t & F^r_t P^m_{t|t} \\ P^m_{t|t} F^{rT}_t & P^m_{t|t} \end{pmatrix} + \begin{pmatrix} G Q_t G^T & 0 \\ 0 & 0 \end{pmatrix} \tag{4.7g}$$

5. Set $t := t + 1$ and repeat from step 1.

4.2.2 Simulation Example

Consider an environment with stationary features according to Figure 4.1a. The robot is assumed to move with constant velocity and constant angular velocity. The positions of the features are measured relative the robot if the distance is less than 1 length unit. The orientation is also measured. Thus, the model can be described as

$$\begin{pmatrix} \mathsf{p}_t \\ \mathsf{v}_t \\ \theta_t \\ \omega_t \end{pmatrix} = \begin{pmatrix} I & TI & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \mathsf{p}_{t-1} \\ \mathsf{v}_{t-1} \\ \theta_{t-1} \\ \omega_{t-1} \end{pmatrix} + \begin{pmatrix} \frac{T^2}{2}I & 0 \\ TI & 0 \\ 0 & \frac{T^2}{2} \\ 0 & T \end{pmatrix} v_{t-1}, \tag{4.8a}$$

$$m_t^i = m_{t-1}^i, \tag{4.8b}$$

$$y_t^{m,i} = R_{be}(\theta_t)(\mathsf{p}_t - m_t^i) + e_t^{m,i}, \quad \text{if } \|\mathsf{p}_t - m_t^i\| \leq 1, \tag{4.8c}$$

$$y_t^\theta = \theta + e_t^\theta, \tag{4.8d}$$

where

$$R_{be}(\theta_t) = \begin{pmatrix} \cos\theta_t & \sin\theta_t \\ -\sin\theta_t & \cos\theta_t \end{pmatrix}, \tag{4.9}$$

p_t is the position, v_t is the velocity, θ_t is the orientation, ω_t is the angular velocity and $T = 1\,\mathrm{s}$.

The SLAM problem is solved using the MATLAB code given in Listing 4.1, which is intended to be pedagogical rather than optimal. The function `meas` and a function to generate data can be found in Appendix C. Run the simulation with the following lines of code:

```
[p theta ym yth]=sim2D;
[xhat P]=EKFslam(ym, yth);
```

Listing 4.1: EKF-SLAM *code:* *`EKFslam.m`*

```
function [xhat P]=EKFslam(ym, yth)
  xhat{1}=[1; 0; -0.0314; 0.2487; 0; 0.25]; % Initial states
  P{1}=0.01*eye(6);  % Initial covariance
  Q=diag([0.1 0.1 0.001]); % Process noise
  Rf=0.01;  % Measurement covariance, features
  Rth=0.01; % Measurement covariance, orientation
  indf=[];  % Index of seen features also order in state vector

  F=[eye(2)   1*eye(2) zeros(2);
     zeros(2) eye(2)   zeros(2);
     zeros(2) zeros(2) [1 1; 0 1]];
  G=[0.5*eye(2) zeros(2,1); 1*eye(2) zeros(2,1);
     zeros(1,2) 0.5; zeros(1,2) 1];

  for t=1:size(ym,2) % Loop over the measurements
    % Measurement update
    [H,ymeas,addf]=meas(ym{t},yth(t),indf,xhat{t});
    K=P{t}*H'*inv(H*P{t}*H'+blkdiag(Rth,Rf*eye(size(H,1)-1)));
    xhat{t}=xhat{t}+K*(ymeas-H*xhat{t});
    P{t}=P{t}-K*H*P{t};
    P{t}=1/2*(P{t}+P{t}'); % Make sure P is symmetric
    Rbe=[cos(xhat{t}(5)) sin(xhat{t}(5));
        -sin(xhat{t}(5)) cos(xhat{t}(5))];
    for i=addf % Add new features to state vector
      xhat{t}=[xhat{t}; xhat{t}(1:2)-Rbe'*ym{t}(:,i)];
      P{t}=blkdiag(P{t},1e6*eye(2));
      indf=[indf i];
    end

    % Time update
    Ftilde=blkdiag(F,eye(size(P{t},2)-6));
    xhat{t+1}=Ftilde*xhat{t};
    P{t+1}=Ftilde*P{t}*Ftilde'+blkdiag(G*Q*G',zeros(2*length(indf)));
  end
end
```

The results of this simulation can be studied in Figure 4.2. It can be seen that the uncertainty about the map and robot position grows with time if the robot is continuously exploring new areas (Figure 4.2b). However, when coming back to a previously mapped feature, the uncertainty is reduced and corrections are made to the map (Figure 4.2c). After some rounds in the loop, a solution close to the true one is obtained (Figure 4.2d).

Compared to the linear problem in (4.3), the only difference in this example is that the orientation (which affects the direction of the map measurements) has to be estimated. The solution to the linear problem was shown in Figure 4.1 and it is clearly visible that this problem is harder. The measurement equation has to be linearized based on a prediction of the orientation. A small error in orientation can give a large error in position if the feature is relatively far away.

4.3 FastSLAM

The approach of FastSLAM is to solve the SLAM problem using a particle filter and was first presented in Montemerlo et al. (2002). The state dimension including both robot states and map states is too high for applying the regular PF. However, making use of the following factorization of the posterior probability

$$p(x^r_{1:t}, m_t | y_{1:t}) = p(x^r_{1:t} | y_{1:t}) p(m_t | x^r_{1:t}, y_{1:t}), \tag{4.10}$$

the RBPF, discussed in Section 3.7, can be used given that the measurements of the map are linear given the robot states. The factorization can be taken one step further,

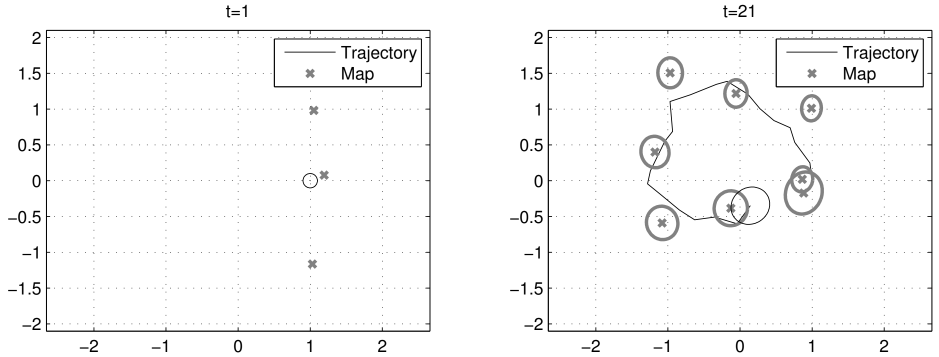
$$p(m_t | x^r_{1:t}, y_{1:t}) = \prod_{i=1}^{N_m} p(m^i_t | x^r_{1:t}, y_{1:t}), \tag{4.11}$$

since all features are independent given the trajectory of the robot. For the SLAM problem, we often have a nonlinear model of the measurements. If it is nonlinear in the map states, the RBPF cannot be directly applied. Linearizing the measurement equation with respect to the map states will then enable the use of RBPF. This gives the following model

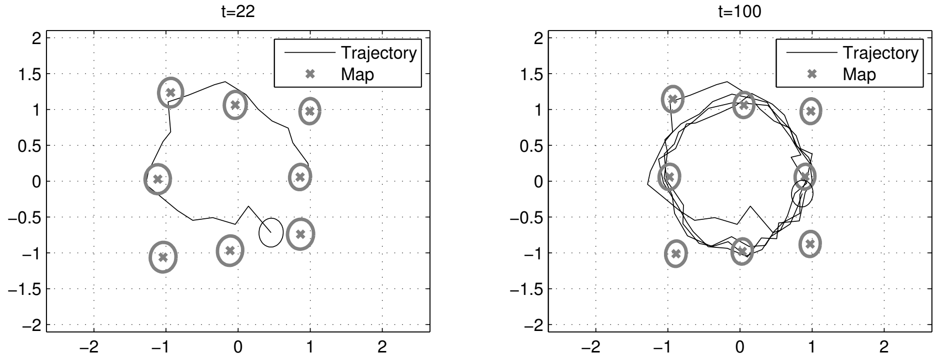
$$x^r_t = f(x^r_{t-1}, u_{t-1}) + v_t, \tag{4.12}$$

$$m^i_t = m^i_{t-1}, \tag{4.13}$$

$$y^i_t \approx h(x^r_t, \hat{m}^i_{t|t-1}) + H^m_t (m^i_t - \hat{m}^i_{t|t-1}) + e_t, \tag{4.14}$$



(a) *Initial map* ***(b)*** *Just before loop closing*



(c) *Just after loop closing* ***(d)*** *After four laps*

Figure 4.2: *Example of the* EKF-SLAM *algorithm. The robot trajectory and the current covariance are plotted together with the mapped features (crosses) and their associated covariance (ellipses).*

where:

$$H_t^m = \left. \frac{\partial h(x_t^r, m_t)}{\partial m_t} \right|_{m_t = \hat{m}_{t|t-1}} \tag{4.15}$$

With this model the RBPF can be applied and then only the robot states need to be treated with the particle filter. The factorization in (4.11) makes it possible to use independent Kalman filters for each mapped feature instead of a combined filter for all features since they are independent. The computational complexity for this algorithm is $\mathcal{O}(N n_m)$, where N is the number of particles and n_m the number of features. By using a tree-based implementation, the complexity can be reduced even further to $\mathcal{O}(N \log n_m)$ which is significantly better than for EKF-SLAM (Montemerlo et al., 2002).

To improve the convergence properties of FastSLAM, an improved proposal distribution was suggested in Montemerlo and Thrun (2007). The new algorithm got the name FastSLAM 2.0 and uses the latest measurements to propose new particles in the time update step. The theory behind FastSLAM is collected in the book Montemerlo and Thrun (2007).

Although the FastSLAM algorithm has proved to work well in practice, there is a degeneration of the map over time. The resampling causes the map associated with the most probable features to be copied and the others to be discarded. For mapped features that have been out of sight for some time, the map will be the same for all particles. This leads to missing information in the cross-correlations for the map and is especially problematic in the case of a loop-closure. This problem is discussed in, e.g., Bailey et al. (2006b).

Traditional implementations of the FastSLAM filter are done for a robot model with low-dimensional state vector, e.g., two-dimensional position and an orientation angle. In order to use FastSLAM for a flying robot, three-dimensional position and three-dimensional orientation must be used. In some cases a dynamic model including velocities, accelerations, and biases is desirable as well. In such a case, the dimension of the state space treated with the particle filter is too big to be computationally feasible. Further factorizations of the posterior in (4.10) must then be done. If the robot state is split as $x_t^r = ({x_t^p}^T \; {x_t^k}^T)^T$, it can be done as

$$p(x_{1:t}^p, x_t^k, m_t | y_{1:t}) = p(x_{1:t}^p | y_{1:t}) p(x_t^k | x_{1:t}^p, y_{1:t}) p(m_t | x_t^k, x_{1:t}^p, y_{1:t}). \tag{4.16}$$

The details of this approach are given in Paper A.

5

Concluding Remarks

THE FIRST PART of this thesis gave the theoretical background for the publications in part two. This chapter first discusses the results of the thesis and then possible future directions of research.

5.1 Results

The results of this thesis all have in common that they can be used for navigation applications. As discussed in the introductory part, estimation and detection are closely related and necessary building blocks in robust robotic applications.

A factorization of the simultaneous localization and mapping problem was provided in Paper A. This factorization made it possible to use high dimensional vehicle models in the particle filter solution to SLAM. The results were applied to data from a helicopter equipped with a vision sensor, an inertial measurement unit and a barometer. For navigation with a vision sensor it is important to detect errors made in the image processing step. It is essential that erroneously associated features are detected as the navigation filter might diverge otherwise. The parity space method is applied to this detection problem in Paper C. If a magnetometer is used to assist in estimating orientation it is important with disturbance monitoring. Paper B presents a parity space formulation of the generalized likelihood ratio test which is applied to monitoring of magnetic disturbances.

In many applications there are sensors with different sampling rate. For example, visual navigation uses an often slow camera and an IMU with high update rate. Paper D provides an efficient PF solution for this situation when the dynamics and the fast sensor are linear with Gaussian noise and the slow sensor is nonlinear with non-Gaussian noise.

Positioning of a ground target with vision sensor is the subject of Paper E. The non-parametric target distribution can represent both the case that the target was not seen in an area and that the target is found. This is an enabler for information theoretic control, where the target always is searched for in areas with high probability.

5.2 Future Work

One of the big challenges in SLAM is how to lower the computational complexity. The nature of the problem is such that a big map gives rise to maintenance of many dependences. Even if the problem could be solved exactly, this would probably lead to high complexity. The question is how to make an approximation that leads to low complexity and at the same time affects the solution as little as possible. Another problem is the ability to do robust association of features. This is especially a problem during loop closure, where the predicted position of the feature might have drifted away from its actual position.

To create a computationally efficient solution to these problems, one way could be to classify the information from the sensors. That is, to classify the objects seen by the sensors as cars, houses, trees and so on. This object oriented framework will probably solve many problems.

Using an object oriented framework, it is easier to understand which features that are important in the image. Take an example with a human driving on a highway. Locally, we remember what we just saw in order to make dead-reckoning possible. But after driving a long distance, only more significant (information rich) things as road crossings, signs and such are remembered. Removing features to lower the complexity was proposed in Dissanayake et al. (2002), but doing this based on the type of object would take SLAM to a new level.

Another area where an object oriented approach would help is for robust feature association. Understanding what the robot sees would help in selecting features that can robustly be found again. Again, compare with a human. We see things around us that we recognize and by experience we can tell how these things are expected to behave. A car is moving and is therefore bad to use for map building, a house has pretty high probability of being stationary and is good for map building and so on.

The particle filter based SLAM methods that are used today have problems with loop closing. This is due to the dependence on the estimate of the trajectory which is poor when using a PF. If the map is conditioned on only the last state, the covariance matrix for the whole map must be calculated which is computer intensive. The advantage is that the cross-correlations in the map are saved which would help during a loop closure. What if the Kalman filter part is represented using the information form? Perhaps the sparse structure can be utilized to make the problem computationally tractable.

The Rao-Blackwellized particle filter was derived for the marginal state in this thesis, opposed to the normal case where the full trajectory of the nonlinear states is estimated. Sampling was still done on a per particle basis. An interesting extension would be to sample from the full distribution as is done for the marginal particle filter (Klaas et al., 2005).

Appendices

A

Probability Distributions

A.1 Gaussian Distribution

The most widely used distribution for random variables is the *Gaussian* or *Normal* distribution. Consider the stochastic variable X which is Gaussian distributed with mean μ and variance σ^2, denoted

$$X \sim \mathsf{N}(\mu,\, \sigma^2).$$

The probability density function, PDF, for X is then

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}. \tag{A.1}$$

This means that the probability for X to have a value in the interval $[a,\, b]$ is

$$P(a \le X \le b) = \int\limits_a^b f_X(x)\, dx.$$

In a more general case, X is an $n \times 1$ vector. The distribution of the Gaussian vector is then represented by the *multivariate Gaussian* PDF

$$f_X(\mathbf{x}) = \frac{1}{(2\pi)^{n/2} \det^{1/2}(S)} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T S^{-1}(\mathbf{x}-\mu)}, \tag{A.2}$$

where μ is the mean vector and S is the positive definite covariance matrix. An example of the PDF for a variable distributed with the parameters $\mu = \begin{pmatrix} 0 & 0 \end{pmatrix}^T$ and $S = \text{diag}(1,\, 2)$ is plotted in Figure A.1.

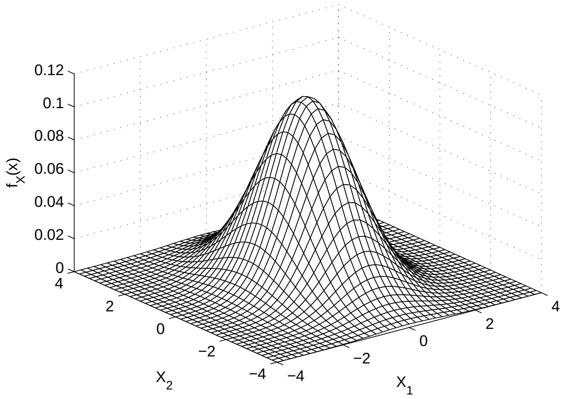


Figure A.1: *The multivariate Gaussian distribution with parameters* $\mu = (0\ \ 0)^T$ *and* $P = \mathit{diag}(1,\,2)$.

A.2 Chi-square Distribution

Squared Gaussian variables are common, especially in detection theory. Often are sums of squared Gaussian residuals used as a test statistic and the result will then be chi-square distributed if the residuals are white and Gaussian. If the residuals have zero mean, the test statistic will have a *central chi-square* distribution. In case of non-zero mean, the distribution will be the *noncentral chi-square*.

A.2.1 Central Chi-square

Sums of squared independent and identically distributed, IID, Gaussian variables with zero mean will become chi-square distributed. That is, $X = \sum_{i=1}^{\nu} X_i^2$ with $X_i \sim \mathsf{N}(0,1)$ will be chi-square distributed with ν degrees of freedom, denoted $X \sim \chi^2_\nu$. The probability density function is

$$f_X(x) = \begin{cases} \frac{1}{2^{\frac{\nu}{2}}\Gamma(\frac{\nu}{2})} x^{\frac{\nu}{2}-1} e^{-\frac{1}{2}x} & x > 0 \\ 0 & x < 0 \end{cases}, \tag{A.3}$$

where $\Gamma(u)$ is the Gamma function, which is defined as

$$\Gamma(u) = \int_0^\infty t^{u-1} e^{-t}\, dt. \tag{A.4}$$

The mean and variance of this distribution are

$$\mathsf{E}\, x = \nu, \tag{A.5}$$

$$\mathsf{Var}(x) = 2\nu. \tag{A.6}$$

A.2.2 Noncentral Chi-square

When taking the sum of squared IID Gaussian random variables with non-zero mean, the distribution of the result will be represented by the *noncentral chi-square distribution*. Consider the case where $X_i \sim \mathsf{N}(\mu_i, 1)$, then the variable $X = \sum_{i=1}^{\nu} X_i^2$ has ν degrees of freedom and the *noncentrality parameter* $\lambda = \sum_{i=1}^{\nu} \mu_i^2$. The distribution of X is denoted $X \sim \chi_\nu'^2(\lambda)$ and the PDF can be expressed as

$$f_X(x) = \begin{cases} \frac{1}{2}\left(\frac{x}{\lambda}\right)^{\frac{\nu-2}{4}} I_{\frac{\nu}{2}-1}(\sqrt{\lambda x})e^{-\frac{1}{2}(x+\lambda)} & x > 0 \\ 0 & x < 0 \end{cases}, \tag{A.7}$$

where $I_r(u)$ is the modified Bessel function of the first kind and order r. This function is defined as

$$I_r(u) = \frac{\left(\frac{1}{2}u\right)^r}{\sqrt{\pi}\Gamma\left(r+\frac{1}{2}\right)} \int\limits_0^\pi \sin^{2r}(\theta) e^{u\cos\theta}\, d\theta. \tag{A.8}$$

The mean and variance of this distribution are

$$\mathsf{E}\, x = \nu + \lambda, \tag{A.9}$$

$$\mathsf{Var}(x) = 2\nu + 4\lambda. \tag{A.10}$$

B

Quaternion Preliminaries

THE QUATERNIONS WERE INVENTED by Sir William Rowan Hamilton as a way to extend the imaginary numbers, see Hamilton (1844). The unit quaternion have later shown to be a useful tool for orientation representations. A comprehensive description of unit quaternions is given by Kuipers (1999) and a nice survey of orientation representations is given by Shuster (1993). A good primer on unit quaternions for orientation representations of airplanes is given by Stevens and Lewis (2003).

This appendix starts with a short review of the mathematics needed for the quaternion derivations. The quaternions are then introduced and the mathematics for rotations and dynamics is then derived.

B.1 Prerequisites in Vector Kinematics and Mathematics

B.1.1 Cross-product

Note the following properties of the cross-product

$$\mathbf{u} \times \mathbf{v} = -\mathbf{v} \times \mathbf{u}, \tag{B.1}$$

$$a(\mathbf{u} \times \mathbf{v}) = (a\mathbf{u}) \times \mathbf{v} = \mathbf{u} \times (a\mathbf{v}), \tag{B.2}$$

$$\mathbf{u} \times (\mathbf{v} + \mathbf{w}) = (\mathbf{u} \times \mathbf{v}) + (\mathbf{u} \times \mathbf{w}), \tag{B.3}$$

$$\mathbf{u} \cdot (\mathbf{v} \times \mathbf{w}) = \mathbf{v} \cdot (\mathbf{w} \times \mathbf{u}) = \mathbf{w} \cdot (\mathbf{u} \times \mathbf{v}), \tag{B.4}$$

$$\mathbf{u} \times (\mathbf{v} \times \mathbf{w}) = \mathbf{v}(\mathbf{w} \cdot \mathbf{u}) - \mathbf{w}(\mathbf{u} \cdot \mathbf{v}). \tag{B.5}$$

Note also that it is possible to write the cross-product as a matrix multiplication

$$\mathbf{u} \times \mathbf{v} = - \begin{pmatrix} 0 & u_3 & -u_2 \\ -u_3 & 0 & u_1 \\ u_2 & -u_1 & 0 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}. \tag{B.6}$$

B.1.2 Vector Rotation

The vector $\mathbf{u}$ is rotated about $\mathbf{n}$ as shown in Figure B.1. The new vector $\mathbf{v}$ can be described as

$$\begin{aligned}\mathbf{v} &= ON + NW + WV = \\ &= (\mathbf{u}\cdot\mathbf{n})\mathbf{n} + \frac{\mathbf{u}-(\mathbf{u}\cdot\mathbf{n})\mathbf{n}}{|\mathbf{u}-(\mathbf{u}\cdot\mathbf{n})\mathbf{n}|}\underbrace{|NV|\cos\mu}_{|NW|} + \underbrace{\frac{\mathbf{u}\times\mathbf{n}}{|\mathbf{u}|\sin\phi}}_{\frac{WV}{|WV|}}\underbrace{|NV|\sin\mu}_{|WV|},\end{aligned} \tag{B.7}$$

where ON denotes the vector that points from O to N and so on. Note that

$$|NV| = |NU| = |\mathbf{u}-(\mathbf{u}\cdot\mathbf{n})\mathbf{n}| = |\mathbf{u}|\sin\phi.$$

Then,

$$\mathbf{v} = (1-\cos\mu)\mathbf{n}(\mathbf{n}\cdot\mathbf{u}) + \cos\mu\mathbf{u} - \sin\mu(\mathbf{n}\times\mathbf{u}), \tag{B.8}$$

which is often referred to as the *rotation formula*.

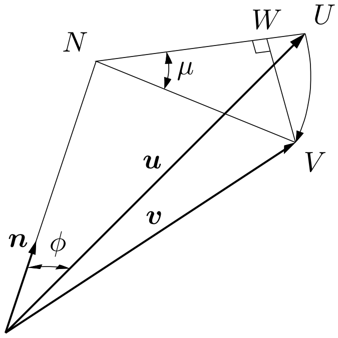


Figure B.1: *The vector* $\mathbf{u}$ *is rotated about* $\mathbf{n}$.

B.1.3 Direction Cosines

To describe the direction of a vector in a unique way, the direction cosines is a useful tool. The angles α, β and γ in Figure B.2 are used. The cosine of the angles will be the projections onto the basis axes. It is possible to describe the direction of $\mathbf{n}$ using only two angles, this is the case when using spherical coordinates as in the right part of Figure B.2. The problem with this description is that a vector pointing in the direction of f_z can have any angle ρ, which is clearly not a unique description.

Observe the following property of the direction cosines. The theorem of Pythagoras can be written

$$|\mathbf{n}|^2\cos^2\alpha + |\mathbf{n}|^2\cos^2\beta + |\mathbf{n}|^2\cos^2\gamma = |\mathbf{n}|^2.$$

This yields that

$$\cos^2\alpha + \cos^2\beta + \cos^2\gamma = 1. \tag{B.9}$$

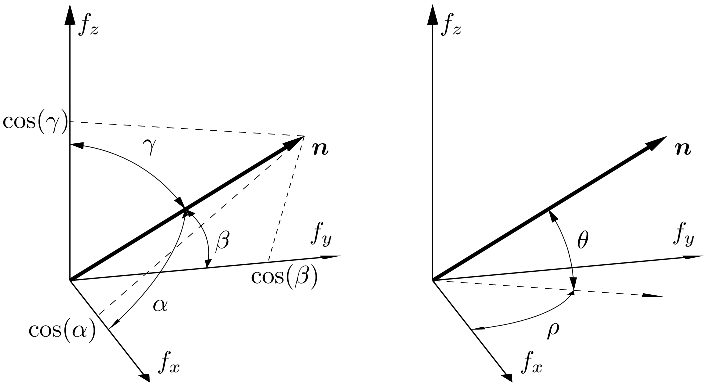


Figure B.2: *Describing the direction of the vector* $\mathbf{n}$ *can be done using the direction angles* α, β, *and* γ *(left), or using only two angles (right).*

B.2 Operations and Properties of Quaternions

The quaternion is a four-tuple with the elements $q_0, \dots, q_3$. It can also be viewed as a vector consisting of the scalar q_0 and the vector $\mathbf{q}$.

$$q = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} q_0 \\ \mathbf{q} \end{pmatrix}. \tag{B.10}$$

Multiplication of two quaternions, denoted with $\odot$, is defined as

$$p \odot q = \begin{pmatrix} p_0 \\ \mathbf{p} \end{pmatrix} \odot \begin{pmatrix} q_0 \\ \mathbf{q} \end{pmatrix} = \begin{pmatrix} p_0 q_0 - \mathbf{p} \cdot \mathbf{q} \\ p_0 \mathbf{q} + q_0 \mathbf{p} + \mathbf{p} \times \mathbf{q} \end{pmatrix}. \tag{B.11}$$

The quaternions have the following properties:

$$p \odot q \neq q \odot p \quad \text{(in general)} \tag{B.12}$$

$$\text{norm}(q) = \sum_{i=0}^{3} q_i^2 \tag{B.13}$$

$$\text{norm}(p \odot q) = \text{norm}(p) \cdot \text{norm}(q) \tag{B.14}$$

$$(p \odot q) \odot r = p \odot (q \odot r) \tag{B.15}$$

$$\tag{B.16}$$

The inverse is defined such that

$$q^{-1} \odot q = q \odot q^{-1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}. \tag{B.17}$$

Thus, the inverse can be calculated as

$$q^{-1} = \begin{pmatrix} q_0 \\ \mathbf{q} \end{pmatrix}^{-1} = \begin{pmatrix} q_0 \\ -\mathbf{q} \end{pmatrix} \Big/ \text{norm}(q) \tag{B.18}$$

B.3 Describing a Rotation with Quaternions

If q is a unit quaternion, that is, $\mathrm{norm}(q) = 1$, it can be written as

$$q = \begin{pmatrix} \cos\delta \\ \sin\delta\mathbf{n} \end{pmatrix}, \tag{B.19}$$

where $\mathbf{n}$ is a unit vector. Hence it can be used to represent a δ rotation about $\mathbf{n}$. The unit vector $\mathbf{n}$ can be described as in Section B.1.3 with the directional cosines

$$\mathbf{n} = \begin{pmatrix} \cos\alpha \\ \cos\beta \\ \cos\gamma \end{pmatrix}. \tag{B.20}$$

Note that

$$\mathrm{norm}(q) = \cos^2\delta + \sin^2\delta(\cos^2\alpha + \cos^2\beta + \cos^2\gamma) = 1, \tag{B.21}$$

which is also a necessity for the quaternion in order to represent a rotation. The variable δ is a measure of the rotation angle as we will see later.

There are two ways of visually describing a rotation. Either a vector is rotated about another vector or the coordinate frame is rotated about a vector, see Figure B.3. When a vector is rotated the transformation gives the new vector coordinates in the frame which remains fixed. When the frame is rotated the vector coordinates are transformed to the new frame. Mathematically, the difference is only the sign of the rotation angle.

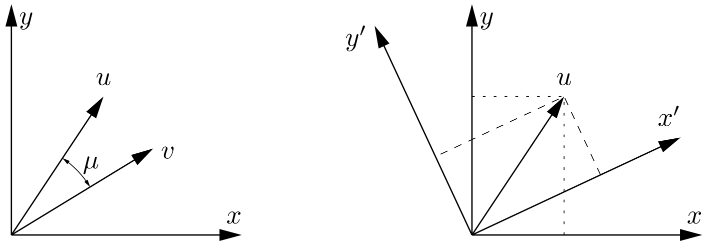


Figure B.3: *In the left figure, a vector is rotated the angle μ counterclockwise. In the right figure, the coordinate frame is rotated the angle μ clockwise.*

To rotate the vector $\mathbf{u}$ about $\mathbf{n}$, (B.8) and Figure B.1 can be used. Observe that the vector rotation is done counterclockwise, which could equally be seen as rotating the coordinate frame clockwise. This rotation can be done using quaternion algebra. If q represents a rotation about $\mathbf{n}$, the rotated vector can be described as

$$v = q^{-1} \odot u \odot q, \tag{B.22}$$

where

$$v = \begin{pmatrix} 0 \\ \mathbf{v} \end{pmatrix} \quad \text{and} \quad u = \begin{pmatrix} 0 \\ \mathbf{u} \end{pmatrix}. \tag{B.23}$$

An alternative description is to use

$$v = q \odot u \odot q^{-1}. \tag{B.24}$$

With this representation vector rotation becomes clockwise and coordinate frame rotation counterclockwise. However, in this work, (B.22) is viewed as the standard rotation.

The result of the quaternion multiplication in (B.22) is

$$v = \begin{pmatrix} \mathbf{q}\cdot\mathbf{u}q_0 - (q_0\mathbf{u} - \mathbf{q}\times\mathbf{u})\cdot\mathbf{q} \\ (\mathbf{q}\cdot\mathbf{u})\mathbf{q} + q_0(q_0\mathbf{u} - \mathbf{q}\times\mathbf{u}) + (q_0\mathbf{u} - \mathbf{q}\times\mathbf{u})\times\mathbf{q} \end{pmatrix}, \tag{B.25}$$

which simplifies to (note the use of (B.5) for the last term in the second row)

$$v = \begin{pmatrix} 0 \\ 2(\mathbf{q}\cdot\mathbf{u})\mathbf{q} + (q_0^2 - \mathbf{q}\cdot\mathbf{q})\mathbf{u} - 2q_0\mathbf{q}\times\mathbf{u} \end{pmatrix}. \tag{B.26}$$

If (B.19) is used, this can be rewritten as

$$v = \begin{pmatrix} 0 \\ 2\sin^2\delta\mathbf{n}(\mathbf{n}\cdot\mathbf{u}) + (\cos^2\delta - \sin^2\delta)\mathbf{u} - 2\cos\delta\sin\delta(\mathbf{n}\times\mathbf{u}) \end{pmatrix}. \tag{B.27}$$

This should be compared with (B.8), since the second row should give the same result. If $\delta = \mu/2$ and some trigonometric identities are applied the results are equal. Thus, inserting $\delta = \mu/2$ in (B.19), the unit quaternion

$$q = \begin{pmatrix} \cos(\mu/2) \\ \sin(\mu/2)\mathbf{n} \end{pmatrix} \tag{B.28}$$

describes a counterclockwise vector rotation or a clockwise coordinate frame rotation about $\mathbf{n}$ with angle μ.

Observe that rotating first with q_a and then with q_b can be done in one step by rotating by $q_a \odot q_b$. Using (B.17) this is easily seen since

$$q_b^{-1} \odot (q_a^{-1} \odot u \odot q_a) \odot q_b = (q_a \odot q_b)^{-1} \odot u \odot (q_a \odot q_b). \tag{B.29}$$

B.4 Rotation Matrix

It is possible to rewrite the quaternion multiplication in (B.22) as a matrix multiplication. Rewriting (B.26) in the form

$$\begin{pmatrix} 0 \\ \mathbf{v} \end{pmatrix} = \begin{pmatrix} 0 \\ R(q)\mathbf{u} \end{pmatrix} \tag{B.30}$$

using (B.6) yields

$$R(q) = \begin{pmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_2 + q_0q_3) & 2(q_1q_3 - q_0q_2) \\ 2(q_1q_2 - q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 + q_0q_1) \\ 2(q_1q_3 + q_0q_2) & 2(q_2q_3 - q_0q_1) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{pmatrix}. \tag{B.31}$$

This is also described in (Kuipers, 1999, p. 158).

B.5 Dynamics of Quaternions

Consider a moving body system. The description of the rotation between the F-system (fixed system) and the B-system (body system) will then contain some dynamics. In order to derive the dynamic equations, the time derivative of the elements of the unit quaternion needs to be calculated.

Let $q_{bf}(t)$ represent the rotation of the B-system with respect to the F-system at time t. Furthermore, let the instantaneous angular velocity of the B-system be in the direction of the unit vector $\hat{\mathbf{s}}^b$, with magnitude ω, which is represented in the B-system. Then, the unit quaternion δq_{bf} describes the rotation about $\hat{\mathbf{s}}^b$ during the small time interval δt. Using small angles approximation and (B.28) this can be written as

$$\delta q_{bf}(\delta t) \approx \begin{pmatrix} 1 \\ \hat{\mathbf{s}}^b \omega \delta t/2 \end{pmatrix}. \tag{B.32}$$

Then, the rotated unit quaternion can be represented as

$$q_{bf}(t+\delta t) = q_{bf}(t) \odot \delta q_{bf}(\delta t). \tag{B.33}$$

The derivative of $q_{bf}(t)$ can then be written as

$$\begin{aligned} \frac{dq_{bf}(t)}{dt} &= \lim_{\delta t \to 0} \frac{q_{bf}(t+\delta t) - q_{bf}(t)}{\delta t} = \lim_{\delta t \to 0} \frac{q_{bf}(t) \odot (\delta q_{bf}(\delta t) - I_q)}{\delta t} = \\ &= \lim_{\delta t \to 0} q_{bf}(t) \odot \left(\frac{\delta q_{bf}(\delta t)}{\delta t} - \frac{I_q}{\delta t} \right) = \frac{1}{2} q_{bf}(t) \odot \begin{pmatrix} 0 \\ \hat{\mathbf{s}}^b \omega \end{pmatrix} = \frac{1}{2} q_{bf}(t) \odot \omega^b_{bf}, \end{aligned} \tag{B.34}$$

where I_q represents the unit quaternion and ω^b_{bf} represents the angular velocity of the B-system relative to the F-system expressed in the B-system. Decompose the unit quaternions as

$$q_{bf} = \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} q_0 \\ \mathbf{q} \end{pmatrix} \quad \text{and} \quad \omega^b_{bf} = \begin{pmatrix} 0 \\ \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} = \begin{pmatrix} 0 \\ \boldsymbol{\omega} \end{pmatrix}. \tag{B.35}$$

Then, the quaternion multiplication in (B.34) can be written as a matrix multiplication with help of (B.11), (B.6) and (B.1).

$$\begin{aligned}
\dot{q}_{bf}(t) = \frac{1}{2} q_{bf}(t) \odot \omega^b_{bf} = \frac{1}{2} \begin{pmatrix} -\mathbf{q} \cdot \boldsymbol{\omega} \\ q_0 \boldsymbol{\omega} + \mathbf{q} \times \boldsymbol{\omega} \end{pmatrix} = \\
= \frac{1}{2} \left(\begin{matrix} -(q_1\omega_x + q_2\omega_y + q_3\omega_z) \\ q_0 \boldsymbol{\omega} - \begin{pmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ q_3 \end{pmatrix} \end{matrix} \right) = \\
= \frac{1}{2} \underbrace{\begin{pmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{pmatrix}}_{S(\omega^b_{bf})} \begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \\
= \frac{1}{2} \underbrace{\begin{pmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{pmatrix}}_{\triangleq S'(q_{bf})} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \qquad \text{(B.36)}
\end{aligned}$$

C

Code for SLAM-Example

The following functions are used to generate the example in Section 4.2.2.

Listing C.1: *`meas.m`*

```
function [H,ymeas,addf]=meas(ym, yth, indf, xhat)
  nf=size(ym,2);         % Total number of features
  H=zeros(1,6+2*length(indf));
  H(5)=1;
  ymeas=yth;            % This vector contains all measurements
  addf=[]; % Contains the indices of features seen for the first time
  Rbe=[cos(xhat(5)) sin(xhat(5)); -sin(xhat(5)) cos(xhat(5))];
  Rdiff=[-sin(xhat(5)) cos(xhat(5)); -cos(xhat(5)) -sin(xhat(5))];
  for i=1:nf
    if ~isnan(ym(1,i)) % Feature is seen at time t
      j=find(indf==i); % Check if feature i has been seen before
      if isempty(j)    % If not, initialize new feature
        addf=[addf i];
      else             % Otherwise, update observation model
        H=[H; Rbe*eye(2) Rdiff*(xhat(1:2)-xhat(6+2*j-1:6+2*j)) ...
           zeros(2,1) zeros(2,2+2*(j-1)) -Rbe*eye(2) ...
           zeros(2,2*(length(indf)-j))];
        ymeas=[ymeas; ym(:,i)];
      end
    end
  end
end
```

Listing C.2: *`sim2D.m`*

```
function [p theta ym yth]=sim2D
  % Build the true map
  m(:,1)=[1 0].’;
  m(:,2)=[1 1].’;
  m(:,3)=[0 1].’;
  m(:,4)=[-1 1].’;
  m(:,5)=[-1 0].';
  m(:,6)=[-1 -1].';
  m(:,7)=[0 -1].';
  m(:,8)=[1 -1].';

  Rf=0.01*eye(2); % Measurement covariance, features
  Rth=0.01;       % Measurement covariance, orientation

  % Build the trajectory
  N=100; % Number of samples
  theta=0:8*pi/N:8*pi;
  p=[cos(theta); sin(theta)];

  for t=1:size(p,2)
    yth(t)=theta(t)+sqrt(Rth)*randn;
    Rbe=[cos(theta(t)) sin(theta(t)); -sin(theta(t)) cos(theta(t))];
    for i=1:size(m,2)
      if norm(p(:,t)-m(:,i))<1.01 % Check if visible
        ym{t}(:,i)=Rbe*(p(:,t)-m(:,i))+sqrtm(Rf)*randn(2,1);
      else
        ym{t}(:,i)=[NaN; NaN];
      end
    end
  end
end
```

Bibliography

M. Abramowitz and I. A. Stegun, editors. *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables.* Dover Publications, 1965.

B. D. O. Anderson and J. B. Moore. *Optimal Filtering.* Prentice-Hall, Inc, Englewood Cliffs, NJ, 1979. ISBN 0-13-638122-7.

C. Andrieu and A. Doucet. Particle filtering for partially observed Gaussian state space models. *Journal of the Royal Statistical Society*, 64(4):827–836, 2002.

M. Arulampalam, S. Maskel, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, Feb. 2002.

T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine*, 13(3):108–117, Sept. 2006.

T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the EKF-SLAM algorithm. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006a.

T. Bailey, J. Nieto, and E. Nebot. Consistency of the FastSLAM algorithm. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, May 2006b.

T. Bayes. An essay towards solving a problem in the doctrine of chances. *The Philosophical Transactions of the Royal Society of London*, 53:370–418, 1763. Reproduced in Bernard and Bayes (1958).

G. A. Bernard and T. Bayes. Studies in the history of probabilty and statistics: IX. Thomas Bayes's essay towards solving a problem in the doctrine of chances. *Biometrika*, 45 (3/4):293–315, Dec. 1958.

Å. Björck. *Numerical Methods for Least Squares Problems.* SIAM Publications, 1996.

R. Chen and J. S. Liu. Mixture Kalman filters. *Journal of the Royal Statistical Society*, 62(3):493–508, 2000.

A. Y. Chow and A. S. Willsky. Analytical redundancy and the design of robust failure detection systems. *IEEE Transactions on Automatic Control*, 29(7):603–614, 1984.

M. Csorba. *Simultaneous Localisation and Map Building.* PhD thesis, University of Oxford, 1997.

A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings Ninth IEEE International Conference on Computer Vision*, volume 2, pages 1403–1410, Nice, France, Oct. 2003.

A. J. Davison, Y. G. Cid, and N. Kita. Real-time 3D SLAM with wide-angle vision. In *Proceedings of the 5th IFAC/EUCON Symposium on Intelligent Autonomus Vehicles*, Lisaboa, Portugal, July 2004.

A. J. Davison, I. Reid, N. Molton, and O. Strasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, June 2007.

P. Del Moral. *Feynman-Kac formulae: Genealogical and Interacting Particle Systems with Applications.* Probability and Applications. Springer, New York, USA, 2004.

G. Dissanayake, S. B. Williams, H. Durrant-Whyte, and T. Bailey. Map management for efficient simultaneous localization and mapping (SLAM). *Autonomous Robots*, 12(3): 267–286, 2002.

M. W. M. G. Dissanyake, P. Newman, S. Clark, and H. F. Durrant-Whyte. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Journal of Robotics and Automation*, 17(3):229–241, June 2001.

A. Doucet. On sequential simulation-based methods for bayesian filtering. Technical Report CUED/F-INFENG/TR.310, Signal Processing Group, Department of Engineering, University of Cambridge, 1998.

A. Doucet and A. M. Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. In D. Crisan and B. Rozovsky, editors, *Handbook of Nonlinear Filtering.* Oxford University Press, 2008. To Appear.

A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice.* Springer Verlag, New York, USA, 2001.

H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping (SLAM): Part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110, June 2006.

H. F. Durrant-Whyte. Uncertain geometry in robotics. *IEEE Journal of Robotics and Automation*, 4(1):23–31, Feb. 1988.

E. Eade and T. Drummond. Scalable monocular SLAM. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 469–476, New York, NY, USA, June 2006.

R. A. Fisher. On the foundations of mathematical statistics. *The Philosophical Transactions of the Royal Society of London*, A(222):309–368, 1922.

J. J. Gertler. *Fault Detection and Diagnosis in Engineering Systems*. Marcel Dekker, Inc., 1998.

G. H. Golub and C. F. van Loan. *Matrix Computations*. John Hopkins University Press, 3 edition, 1996. ISBN 0-2018-54-14-8.

N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings on Radar and Signal Processing*, volume 140, pages 107–113, 1993.

F. Gustafsson. *Adaptive filtering and change detection*. John Wiley & Sons, Ltd, 2 edition, 2001.

S. W. R. Hamilton. On quaternions; or on a new system of imaginaries in algebra. *Philosophical Magazine*, xxv:10–13, July 1844.

C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, Manchester, UK, 1988.

G. Hendeby. *Performance and Implementation Aspects of Nonlinear Filtering*. Dissertations no 1161, Linköping Studies in Science and Technology, Mar. 2008.

G. Hendeby, R. Karlsson, and F. Gustafsson. A new formulation of the Rao-Blackwellized particle filter. In *IEEE Workshop on Statistical Signal Processing*, Madison, WI, USA, Aug. 2007.

Y. C. Ho and R. C. K. Lee. A Bayesian approach to problems in stochastic estimation and control. *IEEE Transactions on Automatic Control*, 9:333–338, Oct. 1964.

D. Howie. *Interpreting Probability: Controversies and Developments in the Early Twentieth Century*. Cambridge University Press, 2002.

X.-L. Hu, T. B. Schön, and L. Ljung. A basic convergence result for particle filtering. *IEEE Transactions on Signal Processing*, 56(4):1337–1348, Apr. 2008.

S. Huang and G. Dissanayake. Convergence and consistency analysis for extended kalman filter based SLAM. *IEEE Transactions on Robotics*, 23(5):1036–1049, Oct. 2007.

A. Isidori. *Nonlinear Control Systems*. Springer-Verlag, 3 edition, 1995.

E. T. Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

P. Jensfelt. *Approaches to Mobile Robot Localization in Indoor Environments*. PhD thesis, Signal, Sensors and Systems (S3), Royal Institute of Technology, SE-100 44 Stockholm, Sweden, 2001.

T. Kailath, A. H. Sayed, and B. Hassibi. *Linear Estimation.* Prentice Hall, 2000.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the American Society of Mechanical Engineering — Journal Basic Engieering*, 82 (Series D):35–45, Mar. 1960.

R. Karlsson and M. Norrlöf. Bayesian position estimation of an industrial robot using multiple sensors. In *IEEE Conference on Control Applications*, Taipei, Taiwan, Sept. 2004.

R. Karlsson, T. B. Schön, D. Törnqvist, G. Conte, and F. Gustafsson. Utilizing model structure for efficient simultaneous localization and mapping for a UAV application. In *Proceedings of IEEE Aerospace Conference*, 2008.

S. M. Kay. *Fundamentals of Statistical Signal Processing: Detection Theory*, volume 2. Prentice-Hall, Inc, 1998. ISBN 0-13-504135-X.

H. K. Khalil. *Nonlinear Systems.* Prentice-Hall, Inc, 3 edition, 2002.

G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

M. Klaas, N. de Freitas, and A. Doucet. Toward practical n^2 monte carlo: the marginal particle filter. In *Proceedings of 21st Conference on Uncertainty in Artificial Intelligence*, Edinburgh, Scotland, 2005.

J. B. Kuipers. *Quaternions and Rotation Sequences.* Princeton Univerity Press, 1999.

S. M. LaValle. *Planning Algorithms.* Cambridge University Press, 2006.

E. L. Lehmann. *Testing Statistical Hypotheses.* Probability and Mathematical Statistics. John Wiley & Sons, Ltd, 2 edition, 1986. ISBN 0-471-84083-1.

J. S. Liu and E. Chen. Sequential Monte Carlo methods for dynamics systems. *Journal of American Statistical Association*, 93:1032–1044, 1998.

N. Metropolis and S. Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

C. Meyer. *Matrix Analysis and Applied Linear Algebra.* SIAM, 2000.

M. Montemerlo and S. Thrun. *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics.* Springer-Verlag, 2007.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM a factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Comference on Artificial Intelligence*, Edmonton, Canada, 2002.

National Geomagnetism Program. Online: http://geomag.usgs.gov, 2008.

J. Neira and J. Tardos. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, Dec. 2001.

J. Neyman and E. S. Pearson. On the use and interpretation of certain test criteria for purposes of statistical inference: Part I. *Biometrika*, 20A(1/2):175–240, July 1928a.

J. Neyman and E. S. Pearson. On the use and interpretation of certain test criteria for purposes of statistical inference: Part II. *Biometrika*, 20A(3/4):263–294, Dec. 1928b.

P.-J. Nordlund. *Sequential Monte Carlo Filters and Integrated Navigation.* Licentiate thesis no 945, Department of Electrical Engineering, Linköpings universitet, Sweden, 2002.

B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter: particle filters for tracking applications.* Artech House, London, UK, 2004.

T. J. Rivlin. *The Chebyshev Polynomials.* John Wiley & Sons, Inc, 1974.

C. P. Robert. *The Bayesian Choice: From Decision-Theoretic Foundation to Computational Implementation.* Springer texts in Statistics. Springer-Verlag, 2 edition, 2001.

W. J. Rugh. *Linear System Theory.* Prentice-Hall, Inc, 1996. ISBN 0-13-441205-2.

F. J. Samaniego and D. M. Reneau. Toward a reconciliation of the bayesian and frequentist approaches to point estimation. *Journal of the American Statistical Association*, 89 (427):947–957, Sept. 1994.

T. Schön, F. Gustafsson, and P.-J. Nordlund. Marginalized particle filters for mixed linear/nonlinear state-space models. *IEEE Transactions on Signal Processing*, 53(7): 2279–2289, July 2005.

T. Schön, R. Karlsson, D. Törnqvist, and F. Gustafsson. A framework for simultaneous localization and mapping utilizing model structure. In *The 10th International Conference on Information Fusion*, Quebec, Canada, Aug. 2007a.

T. Schön, D. Törnqvist, and F. Gustafsson. Fast particle filters for multi-rate sensors. In *The 15th European Signal Processing Conference (EUSIPCO 2007)*, Poznan, Poland, Sept. 2007b.

T. B. Schön. *Estimation of Nonlinear Dynamic Systems: Theory and Applications.* Dissertations no 998, Department of Electrical Engineering, Linköpings universitet, Sweden, SE-581 83 Linköping, Sweden, Feb. 2006.

M. D. Shuster. A survey of attitude representations. *The Journal of Astronautical Sciences*, 41(4):439–517, 1993.

R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I. J. Cox and G. T. Wilfong, editors, *Autonomous robot vehicles*, pages 167–193. Springer-Verlag, 1990.

B. L. Stevens and F. L. Lewis. *Aircraft control and simulation.* John Wiley & Sons, Inc., 2 edition, 2003.

S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 23(7–8):693–716, 2004.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics.* The MIT Press, 2005.

J. Tisdale, A. Ryan, Z. Kim, D. Törnqvist, and J. K. Hedrick. A multiple UAV system for vision-based search and localization. In *American Control Conference*, Seattle, WA, USA, June 2008.

D. Törnqvist. *Statistical Fault Detection with Applications to IMU Disturbances.* Licentiate thesis no. 1258, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, June 2006.

D. Törnqvist and F. Gustafsson. Eliminating the initial state for the generalized likelihood ratio test. In *Proceedings of IFAC Symposium SAFEPROCESS*, Beijing, China, Aug. 2006.

D. Törnqvist, F. Gustafsson, and I. Klein. GLR tests for fault detection over sliding data windows. In *Proceedings of the 16th IFAC World Congress*, Prague, Czech Republic, July 2005.

D. Törnqvist and F. Gustafsson. Unifying the parity-space and GLR approach to fault detection with an IMU application. *Automatica*, 2008. Submitted.

D. Törnqvist, T. B. Schön, and F. Gustafsson. Detecting spurious features using parity space. In *Proceedings of 10th International Conference on Control, Automation, Robotics and Vision*, Hanoi, Vietnam, Dec. 2008a. Accepted for publication.

D. Törnqvist, T. B. Schön, R. Karlsson, and F. Gustafsson. Particle filter SLAM with high dimensional vehicle model. *Journal of Intelligent and Robotic Systems*, 2008b. Accepted for publication.

C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. l. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, Aug. 2008.

V. Šmídl and A. Quinn. *The Variational Bayes Method in Signal Processing.* Springer-Verlag, 2006.

Xsens Technologies B.V. Online: http://www.xsens.com, 2008.

Part II

Publications

Paper A

Particle Filter SLAM with High Dimensional Vehicle Model

Authors: David Törnqvist, Thomas B. Schön, Rickard Karlsson and Fredrik Gustafsson

Edited version of the paper: D. Törnqvist, T. B. Schön, R. Karlsson, and F. Gustafsson. Particle filter SLAM with high dimensional vehicle model. *Journal of Intelligent and Robotic Systems*, 2008b. Accepted for publication

Parts of the paper in:

> R. Karlsson, T. B. Schön, D. Törnqvist, G. Conte, and F. Gustafsson. Utilizing model structure for efficient simultaneous localization and mapping for a UAV application. In *Proceedings of IEEE Aerospace Conference*, 2008.
>
> T. Schön, R. Karlsson, D. Törnqvist, and F. Gustafsson. A framework for simultaneous localization and mapping utilizing model structure. In *The 10th International Conference on Information Fusion*, Quebec, Canada, Aug. 2007a.

Preliminary version: Published as Technical Report LiTH-ISY-R-2863, Dept. of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

Particle Filter SLAM with High Dimensional Vehicle Model

David Törnqvist, Thomas B. Schön, Rickard Karlsson and Fredrik Gustafsson

Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
{tornqvist, schon, rickard, fredrik}@isy.liu.se

Abstract

This work presents a particle filter (PF) method closely related to FastSLAM for solving the simultaneous localization and mapping (SLAM) problem. Using the standard FastSLAM algorithm, only low-dimensional vehicle models can be handled due to computational constraints. In this work an extra factorization of the problem is introduced that makes high-dimensional vehicle models computationally feasible. Results using experimental data from a UAV (helicopter) are presented. The algorithm fuses measurements from on-board inertial sensors (accelerometer and gyro), barometer, and vision in order to solve the SLAM problem.

1 Introduction

The main task in localization/positioning and tracking is to estimate, for instance, the position and orientation of the object under consideration. The *particle filter* (PF), (Gordon et al., 1993, Doucet et al., 2001a), has proved to be an enabling technology for many applications of this kind, in particular when the observations are complicated nonlinear functions of the position and heading (Gustafsson et al., 2002). Furthermore, the *Rao-Blackwellized particle filter* (RBPF) also denoted the *marginalized particle filter* (MPF), (Doucet et al., 2000, Casella and Robert, 1996, Doucet et al., 2001b, Chen and Liu, 2000, Andrieu and Doucet, 2002, Schön et al., 2005) enables estimation of velocity, acceleration, and sensor error models by utilizing any linear Gaussian sub-structure in the model, which is fundamental for performance in applications as surveyed in Schön et al. (2006). As described in Schön et al. (2005), the RBPF splits the state vector x_t into two parts, one part x_t^p which is estimated using the particle filter and another part x_t^k where Kalman filters are applied. Basically, it uses the following factorization of the posterior distribution

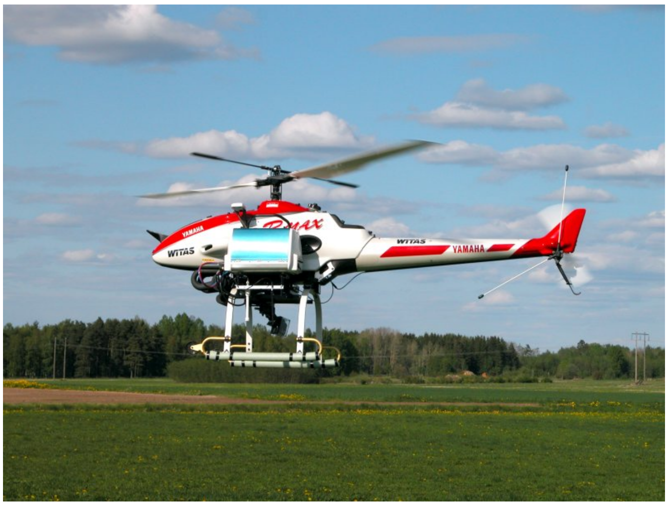


Figure 1: *The Yamaha RMAX helicopter used in the experiments. The on-board system is equipped with an* IMU *sensor (accelerometer and gyro) and a vision sensor. The on-board* GPS *receiver is used for evaluation only.*

of the state vector, which follows from Bayes' rule,

$$p(x^p_{1:t}, x^k_t | y_{1:t}) = p(x^k_t | x^p_{1:t}, y_{1:t}) p(x^p_{1:t} | y_{1:t}), \tag{1}$$

where $y_{1:t} \triangleq \{y_1, \ldots, y_t\}$ denotes the measurements up to time t. If the model is conditionally linear Gaussian, i.e., if the term $p(x^k_t | x^p_{1:t}, y_{1:t})$ is linear Gaussian, it can be optimally estimated using the Kalman filter, whereas for the second factor we have to resort to the PF.

Simultaneous localization and mapping (SLAM) is an extension of the localization or positioning problem to the case where the environment is unmodeled and has to be mapped on-line. An introduction to the SLAM problem is given in the survey papers Bailey and Durrant-Whyte (2006), Durrant-Whyte and Bailey (2006) and the recent book Thrun et al. (2005). From a sensor point of view, there are two ways of tackling this problem. The first way is to use only one sensor, such as vision, see e.g., Davison et al. (2007), Davison (2003), Davison et al. (2004), Eade and Drummond (2006) and the second way is to fuse measurements from several sensors. This work considers the latter. The FastSLAM algorithm introduced in Montemerlo et al. (2002) has proved to be an enabling technology for such applications. FastSLAM can be seen as a special case of RBPF/MPF, where the map state m_t, containing the positions for all landmarks used in the mapping, can be interpreted as a linear Gaussian state. The main difference is that the map vector is a constant parameter with a dimension increasing over time, rather than a time-varying state with a dynamic evolution over time. The derivation is completely analogous to (1), and makes use of the following factorization

$$p(x_{1:t}, m_t | y_{1:t}) = p(m_t | x_{1:t}, y_{1:t}) p(x_{1:t} | y_{1:t}). \tag{2}$$

The FastSLAM algorithm was originally devised to solve the SLAM problem for mobile robots, where the dimension of the state vector is small, typically consisting of three states (2D position and a heading angle) (Thrun et al., 2005). This implies that all platform states must be estimated by the PF, which is computationally demanding for high order models.

Paralleling the evolution of PF applications to high dimensional state vectors, the aim of this contribution is to build on our earlier work (Schön et al., 2007a) which unify the ideas presented in Schön et al. (2005) and Schön et al. (2007b). This is done in order to extend the FastSLAM algorithm (Montemerlo et al., 2002) to be able to cope with high dimensional state vectors as well. Basically, the main result follows from

$$p(x^p_{1:t}, x^k_t, m_t|y_{1:t}) = p(m_t|x^k_t, x^p_{1:t}, y_{1:t})p(x^k_t|x^p_{1:t}, y_{1:t})p(x^p_{1:t}|y_{1:t}). \tag{3}$$

The derived algorithm is applied to experimental data from an autonomous aerial vehicle using the RMAX helicopter platform (Figure 1) although the framework is general and could be applied to other platforms as well. The main navigation sensor unit consists of three accelerometers, three gyros, a pressure sensor, and a camera. GPS is used only for evaluation purposes.

In Section 2 the problem under consideration is formulated in more detail. The proposed algorithm is given and explained in Section 3. This algorithm is then applied to an application example in Section 4. Finally, the conclusions are given in Section 5.

2 Problem Formulation

The aim of this work is to solve the SLAM problem when the state dimension of the platform (UAV) is too large to be estimated by the PF. This section provides a more precise problem formulation and introduces the necessary notation.

The total state vector to be estimated at time t is

$$x_t = \begin{pmatrix}(x^p_t)^T & (x^k_t)^T & m^T_t\end{pmatrix}^T, \tag{4}$$

where x^p_t denotes the states of the platform that are estimated by the particle filter, and x^k_t denotes the states of the platform that are linear-Gaussian given information about x^p_t. These states together with the map (landmarks) m_t are estimated using Kalman filters. The map states m_t consists of the entire map at time t, i.e.,

$$m_t = \begin{pmatrix}m^T_{1,t} & \dots & m^T_{M_t,t}\end{pmatrix}^T, \tag{5}$$

where $m_{j,t}$ denotes the position of the j^{th} map entry and M_t denotes the number of entries in the map at time t.

The aim of this work can be formalized as trying to estimate the following filtering *probability density function* (PDF),

$$p(x^p_t, x^k_t, m_t|y_{1:t}). \tag{6}$$

In other words, we are trying to solve the nonlinear filtering problem, providing an estimate of (6). The *key* factorization, which allows us to solve this problem successfully is

$$p(x^p_{1:t}, x^k_t, m_t|y_{1:t}) = \prod_{j=1}^{M_t} \underbrace{p(m_{j,t}|x^p_{1:t}, x^k_t, y_{1:t}) p(x^k_t|x^p_{1:t}, y_{1:t})}_{\text{(extended) Kalman filter}} \underbrace{p(x^p_{1:t}|y_{1:t})}_{\text{particle filter}} \tag{7}$$

In order to devise an estimator for (6) a *system model* and a *measurement model* are needed. The former describes the dynamic behavior of the platform, that is how the state x_t evolves over time. The measurement model describes the sensors, i.e., it consists of equations relating the measurements y_t to the state x_t. We want a general algorithm, which is applicable to many different platforms (aircraft, helicopters, cars, etc.). Hence, the model structure should be as general as possible,

$$x^p_{t+1} = f^p_t(x^p_t) + A^p_t(x^p_t)x^k_t + G^p_t(x^p_t)w^p_t, \tag{8a}$$

$$x^k_{t+1} = f^k_t(x^p_t) + A^k_t(x^p_t)x^k_t + G^k_t(x^p_t)w^k_t, \tag{8b}$$

$$m_{j,t+1} = m_{j,t}, \tag{8c}$$

$$y_{1,t} = h_{1,t}(x^p_t) + C_{1,t}(x^p_t)x^k_t + e_{1,t}, \tag{8d}$$

$$y^{(j)}_{2,t} = h_{2,t}(x^p_t) + H_{j,t}(x^p_t)m_{j,t} + e^{(j)}_{2,t}, \tag{8e}$$

where $j = 1, \dots, M_t$ and the noise for the platform states is assumed white and Gaussian distributed with

$$w_t = \begin{pmatrix} w^p_t \\ w^k_t \end{pmatrix} \sim \mathcal{N}(0, Q_t), \quad Q_t = \begin{pmatrix} Q^p_t & Q^{pk}_t \\ (Q^{pk}_t)^T & Q^k_t \end{pmatrix}. \tag{8f}$$

To simplify the notation in the rest of the paper, denote $f^p_t(x^p_t)$ with f^p_t, $A^p_t(x^p_t)$ with A^p_t and so on. The measurement noise is assumed white and Gaussian distributed according to

$$e_{1,t} \sim \mathcal{N}(0, R_{1,t}), \tag{8g}$$

$$e^{(j)}_{2,t} \sim \mathcal{N}(0, R^j_{2,t}), \qquad j = 1, \dots, M_t. \tag{8h}$$

Finally, x^k_0 is Gaussian,

$$x^k_0 \sim \mathcal{N}(\bar{x}_0, \bar{P}_0), \tag{8i}$$

and the density for x^p_0 can be arbitrary, but it is assumed known.

There are two different measurement models, (8d) and (8e), where the former only models quantities related to the platform, whereas the latter also involve the map states. Section 4 describes a detailed application example using experimental data, where (8d) is used to model inertial sensors and a pressure sensor and (8e) is used to model a camera.

3 Particle Filter for SLAM Utilizing Structure

This section is devoted to deriving and explaining the proposed SLAM algorithm on a rather detailed level. However, when we make use of standard results we just provide the necessary references. The algorithm is given first and the steps are then discussed in more detail.

3.1 Algorithm

The algorithm presented in this paper draws on several rather well known algorithms. It is based on the RBPF/MPF method (Doucet et al., 2000, Casella and Robert, 1996, Doucet et al., 2001b, Chen and Liu, 2000, Andrieu and Doucet, 2002, Schön et al., 2005). The FastSLAM algorithm (Montemerlo et al., 2002) is extended by not only including the map states in the linear part, but also the states corresponding to a linear Gaussian sub-structure present in the model for the platform. Assuming that the platform is modeled in the form (8), the SLAM algorithm utilizing structure is given in Algorithm 1.

Note that y_t denotes the measurements present at time t. The following theorem will give all the details for how to compute the Kalman filtering quantities. It is important to stress that all embellishments available for the particle filter can straightforwardly incorporated into Algorithm 1. To give one example, the so-called FastSLAM 2.0 makes use of an improved proposal distribution in step 6b (Montemerlo et al., 2003).

Theorem 1
Using the model given in (8)*, the conditional probability density functions for* x_t^k *and* x_{t+1}^k *are given by*

$$p(x_t^k | x_{1:t}^p, y_{1:t}) = \mathcal{N}(\hat{x}_{t|t}^k, P_{t|t}), \tag{10a}$$

$$p(x_{t+1}^k | x_{1:t+1}^p, y_{1:t}) = \mathcal{N}(\hat{x}_{t+1|t}^k, P_{t+1|t}), \tag{10b}$$

where

$$\hat{x}_{t|t}^k = \hat{x}_{t|t-1}^k + K_t(y_{1,t} - h_{1,t} - C_t \hat{x}_{t|t-1}^k), \tag{11a}$$

$$P_{t|t} = P_{t|t-1} - K_t S_{1,t} K_t^T, \tag{11b}$$

$$S_{1,t} = C_t P_{t|t-1} C_t^T + R_{1,t}, \tag{11c}$$

$$K_t = P_{t|t-1} C_t^T S_{1,t}^{-1}, \tag{11d}$$

and

$$\begin{aligned}
\hat{x}_{t+1|t}^k &= \bar{A}_t^k \hat{x}_{t|t}^k + G_t^k (Q_t^{kp})^T (G_t^p Q_t^p)^{-1} z_t + f_t^k + L_t(z_t - A_t^p \hat{x}_{t|t}^k), \\
P_{t+1|t} &= \bar{A}_t^k P_{t|t} (\bar{A}_t^k)^T + G_t^k \bar{Q}_t^k (G_t^k)^T - L_t S_{2,t} L_t^T, && (12a) \\
S_{2,t} &= A_t^p P_{t|t} (A_t^p)^T + G_t^p Q_t^p (G_t^p)^T, && (12b) \\
L_t &= \bar{A}_t^k P_{t|t} (A_t^p)^T S_{2,t}^{-1}, && (12c)
\end{aligned}$$

where

$$z_t = x_{t+1}^p - f_t^p, \tag{13a}$$

$$\bar{A}_t^k = A_t^k - G_t^k (Q_t^{kp})^T (G_t^p Q_t^p)^{-1} A_t^p, \tag{13b}$$

$$\bar{Q}_t^k = Q_t^k - (Q_t^{kp})^T (Q_t^p)^{-1} Q_t^{kp}. \tag{13c}$$

Proof: The proof is derived in Schön et al. (2005) for the case without map features, but with linear Gaussian dynamics as a sub-structure. The extension, including the linear Gaussian map sub-structure, falls within the same framework. □

Algorithm 1 Particle filter for SLAM utilizing structure

1. Initialize N particles according to

$$\begin{aligned} x_{1|0}^{p,(i)} &\sim p(x_{1|0}^{p}), \\ x_{1|0}^{k,(i)} &= \bar{x}_{1|0}^{k}, \\ P_{1|0}^{k,(i)} &= \bar{P}_{1|0}, \qquad i = 1, \ldots, N. \end{aligned}$$

2. If there are new map related measurements available, compute the necessary correspondences to the existing states, otherwise proceed to step 3.

3. Compute the importance weights according to

$$\gamma_t^{(i)} = p(y_t | x_{1:t}^{p,(i)}, y_{1:t-1}), \qquad i = 1, \ldots, N,$$

   and normalize $\tilde{\gamma}_t^{(i)} = \gamma_t^{(i)} / \sum_{j=1}^{N} \gamma_t^{(j)}$.

4. Draw N new particles with replacement (resampling) according to, for each $i = 1, \ldots, N$

$$\Pr(x_{t|t}^{(i)} = x_{t|t}^{(j)}) = \tilde{\gamma}_t^{(j)}, \qquad j = 1, \ldots, N.$$

5. If there is a new map related measurement, perform map estimation and management (detailed below), otherwise proceed to step 6.

6. Particle filter prediction and Kalman filter (for each particle $i = 1, \ldots, N$)
   (a) Kalman filter measurement update,

$$p(x_t^k | x_{1:t}^p, y_{1:t}) = \mathcal{N}(x_t^k | \hat{x}_{t|t}^{k,(i)}, P_{t|t}^{(i)}),$$

   where $\hat{x}_{t|t}^{k,(i)}$ and $P_{t|t}^{(i)}$ are given in (11).

   (b) Time update for the nonlinear state,

$$x_{t+1|t}^{p,(i)} \sim p(x_{t+1} | x_{1:t}^{p,(i)}, y_{1:t}).$$

   (c) Kalman filter time update,

$$p(x_{t+1}^k | x_{1:t+1}^p, y_{1:t}) = \mathcal{N}(x_{t+1|t}^k | \hat{x}_{t+1|t}^{k,(i)}, P_{t+1|t}^{(i)}),$$

   where $\hat{x}_{t+1|t}^{k,(i)}$ and $P_{t+1|t}^{(i)}$ are given by (12).

7. Set $t := t + 1$ and repeat from step 2.

3.2 Likelihood Computation

In order to compute the importance weights $\{\gamma_t^{(i)}\}_{i=1}^N$ in Algorithm 1, the following likelihoods have to be evaluated

$$\gamma_t^{(i)} = p(y_t|x_{1:t}^{p,(i)}, y_{1:t-1}), \qquad i = 1, \dots, N. \tag{14}$$

The standard way of performing this type of computation is simply to marginalize the Kalman filter variables x_t^k and $\{m_{j,t}\}_{j=1}^{M_t}$ according to

$$p(y_t|x_{1:t}^{p,(i)}, y_{1:t-1}) = \int p(y_t, x_t^k, m_t|x_{1:t}^{p,(i)}, y_{1:t-1}) dx_t^k dm_t, \tag{15}$$

where

$$\begin{aligned} p(y_t, x_t^k, m_t|x_{1:t}^{p,(i)}, y_{1:t-1}) = p(y_t|x_t^k, m_t, x_t^{p,(i)}) \times& \\ p(x_t^k|x_{1:t}^{p,(i)}, y_{1:t-1}) \prod_{j=1}^{M_t} p(m_{j,t}|x_{1:t}^{p,(i)}, y_{1:t-1}).& \end{aligned} \tag{16}$$

Let us consider the case where both $y_{1,t}$ and $y_{2,t}$ are present, i.e., $y_t = \begin{pmatrix} y_{1,t}^T & y_{2,t}^T \end{pmatrix}^T$. Note that the cases where either $y_{1,t}$ or $y_{2,t}$ are present are obviously special cases. First of all, the measurements are conditionally independent given the state, implying that

$$p(y_t|x_t^k, m_t, x_t^{p,(i)}) = p(y_{1,t}|x_t^k, x_t^{p,(i)}) \prod_{j=1}^{M_t} p(y_{2,t}^{(j)}|x_t^{p,(i)}, m_{j,t}). \tag{17}$$

Now, inserting (17) into (16) gives

$$\begin{aligned} p(y_t, x_t^k, m_t|x_{1:t}^{p,(i)}, y_{1:t-1}) = p(y_{1,t}|x_t^k, x_t^{p,(i)}) p(x_t^k|x_{1:t}^{p,(i)}, y_{1:t-1}) \times& \\ \prod_{j=1}^{M_t} p(m_{j,t}|x_{1:t}^{p,(i)}, y_{1:t-1}) p(y_{2,t}^{(j)}|x_t^{p,(i)}, m_{j,t}),& \end{aligned} \tag{18}$$

which inserted in (15) finally results in

$$\begin{aligned} p(y_t|x_{1:t}^{p,(i)}, y_{1:t-1}) = \int p(y_{1,t}|x_t^k, x_t^{p,(i)}) p(x_t^k|x_{1:t}^{p,(i)}, y_{1:t-1}) dx_t^k& \\ \times \prod_{j=1}^{M_t} \int p(y_{2,t}^{(j)}|x_t^{p,(i)}, m_{j,t}) p(m_{j,t}|x_{1:t}^{p,(i)}, y_{1:t-1}) dm_{1,t} \cdots dm_{M_t,t}.& \end{aligned} \tag{19}$$

All the densities present in (19) are known according to

$$p(x_t^k|x_{1:t}^p, y_{1:t-1}) = \mathcal{N}(x_t^k|\hat{x}_{t|t-1}^k, P_{t|t-1}), \tag{20a}$$

$$p(m_{j,t}|x_{1:t}^p, y_{1:t-1}) = \mathcal{N}(m_{j,t}|\hat{m}_{j,t-1}, \Sigma_{j,t-1}), \tag{20b}$$

$$p(y_{1,t}|x_t^k, x_t^p) = \mathcal{N}(y_{1,t}|h_{1,t} + C_t x_t^k, R_1), \tag{20c}$$

$$p(y_{2,t}^{(j)}|x_t^p, m_{j,t}) = \mathcal{N}(y_{2,t}^{(j)}|h_{2,t} + H_{j,t} m_{j,t}, R_2^j). \tag{20d}$$

Here it is important to note that the standard FastSLAM approximation (linearization) has been invoked in order to obtain (20d). That is, the measurement equation often has to be linearized with respect to the map states $m_{j,t}$ in order to comply with (8e). The reason for this approximation is that we want to use a model suitable for the RBPF/MPF, otherwise the dimension will be much too large for the particle filter to handle. Using (20), the integrals in (19) can now be solved, resulting in

$$\begin{aligned} p(y_t|x^{p,(i)}_{1:t}, y_{1:t-1}) &= \mathcal{N}(y_{1,t}|h_{1,t} + C_t\hat{x}^{k,(i)}_{t|t-1}, C_t P^{(i)}_{t|t-1} C_t^T + R_1) \\ &\times \prod_{j=1}^{M_t} \mathcal{N}(y^{(j)}_{2,t}|h_{2,t} + H_{j,t}\hat{m}_{j,t-1}, H_{j,t}\Sigma_{j,t-1}(H_{j,t})^T + R^j_2). \end{aligned} \tag{21}$$

3.3 Map Estimation and Map Management

A simple map consists of a collection of map point entries $\{m_{j,t}\}_{j=1}^{M_t}$, each parameterized by:

- $\hat{m}_{j,t}$ – estimate of the position (three dimensions).
- $\Sigma_{j,t}$ – covariance for the position estimate.

Note that this is a very simple map parametrization. Each particle has an entire map estimate associated to it. Step 5 of Algorithm 1 consists of updating these map estimates in accordance with the new map-related measurements that are available. First of all, if a measurement has been successfully associated to a certain map entry, it is updated using the standard Kalman filter measurement update according to

$$m_{j,t} = m_{j,t-1} + K_{j,t}\left(y^{(j)}_{2,t} - h_{2,t}\right), \tag{22a}$$

$$\Sigma_{j,t} = \left(I - K_{j,t}H^T_{j,t}\right)\Sigma_{j,t-1}, \tag{22b}$$

$$K_{j,t} = \Sigma_{j,t-1}H^T_{j,t}\left(H_{j,t}\Sigma_{j,t-1}H^T_{j,t} + R_2\right)^{-1}. \tag{22c}$$

If an existing map entry is not observed, the corresponding map estimate is simply propagated according to its dynamics, i.e., it is unchanged

$$m_{j,t} = m_{j,t-1}, \tag{23a}$$

$$\Sigma_{j,t} = \Sigma_{j,t-1}. \tag{23b}$$

Finally, initialization of new map entries have to be handled. If $h_{2,t}(x^p_t, m_{j,t})$ is bijective with respect to the map $m_{j,t}$ this can be used to directly initialize the position from the measurement $y_{2,t}$. However, this is typically not the case, implying that we cannot uniquely initialize the position of the corresponding map entry. This can be handled in different ways. In Section 4 the vision sensor and different techniques are briefly discussed.

4 Application Example

In this section we provide a description of the SLAM application, where Algorithm 1 is used to fuse measurements from a camera, three accelerometers, three gyros and an air-pressure sensor. The sensors are mounted on the RMAX helicopter shown previously in Figure 1. The main objective is to find the position and orientation of the sensor from sensor data only, despite problems such as biases in the measurements. The vision system can extract and track features that are used in SLAM to reduce the inertial drift and the bias in the IMU sensor.

4.1 Model

The basic part of the state vector consists of position $p_t \in \mathbb{R}^3$, velocity $v_t \in \mathbb{R}^3$, and acceleration $a_t \in \mathbb{R}^3$, all described in an earth-fixed reference frame. Furthermore, the state vector is extended with bias states for acceleration $b_{a,t} \in \mathbb{R}^3$, and angular velocity $b_{\omega,t} \in \mathbb{R}^3$ in order to account for sensor imperfections. The state vector also contains the angular velocity ω_t and a unit quaternion q_t, where the latter is used to parametrize the orientation of the UAV.

In order to relate the model to the RBPF/MPF framework, the state vector is split into two parts, one estimated using Kalman filters x_t^k and one estimated using the particle filter x_t^p. Hence, define

$$x_t^k = \begin{pmatrix} v_t^T & a_t^T & (b_{\omega,t})^T & (b_{a,t})^T & \omega_t^T \end{pmatrix}^T, \tag{24a}$$

$$x_t^p = \begin{pmatrix} p_t^T & q_t^T \end{pmatrix}^T, \tag{24b}$$

which means $x_t^k \in \mathbb{R}^{15}$ and $x_t^p \in \mathbb{R}^7$. In inertial estimation it is essential to clearly state which coordinate frame an entity is expressed in. Here the notation is simplified by suppressing the coordinate frame for the earth-fixed states, which means that

$$p_t = p_t^e, \qquad v_t = v_t^e, \qquad a_t = a_t^e, \tag{25a}$$

$$\omega_t = \omega_t^b, \qquad b_{\omega,t} = b_{\omega,t}^b, \qquad b_{a,t} = b_{a,t}^b. \tag{25b}$$

Likewise, the unit quaternions represent the rotation from the earth-fixed system to the body (IMU) system, since the IMU is rigidly attached to the body (strap-down),

$$q = q^{be} = \begin{pmatrix} q_0 & q_1 & q_2 & q_3 \end{pmatrix}^T, \tag{26}$$

where time indices are omitted. The quaternion estimates are normalized, to make sure that they still parametrize an orientation. Further details regarding orientation and coordinate systems are given in Appendix A.1.

Dynamic Model

The dynamic model describes how the platform and the map evolve over time. These equations are given below, in the form (8a) – (8d), suitable for direct use in Algorithm 1.

$$\underbrace{\begin{pmatrix} v_{t+1} \\ a_{t+1} \\ b_{\omega,t+1} \\ b_{a,t+1} \\ \omega_{t+1} \end{pmatrix}}_{x^k_{t+1}} = \underbrace{\begin{pmatrix} I & TI & 0 & 0 & 0 \\ 0 & I & 0 & 0 & 0 \\ 0 & 0 & I & 0 & 0 \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{pmatrix}}_{A^k_t} \underbrace{\begin{pmatrix} v_t \\ a_t \\ b_{\omega,t} \\ b_{a,t} \\ \omega_t \end{pmatrix}}_{x^k_t} + \underbrace{\begin{pmatrix} \frac{T^2}{2} & 0 & 0 & 0 \\ TI & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix}}_{G^k_t} \underbrace{\begin{pmatrix} w_{1,t} \\ w_{2,t} \\ w_{3,t} \\ w_{4,t} \end{pmatrix}}_{w^k_t} \tag{27a}$$

$$\underbrace{\begin{pmatrix} p_{t+1} \\ q_{t+1} \end{pmatrix}}_{x^p_{t+1}} = \underbrace{\begin{pmatrix} p_t \\ q_t \end{pmatrix}}_{f^p_t(x^p_t)} + \underbrace{\begin{pmatrix} TI & \frac{T^2}{2}I & 0_{3\times 9} \\ 0_{4\times 3} & 0_{4\times 9} & \frac{T}{2}\tilde{S}(q_t) \end{pmatrix}}_{A^p_t(x^p_t)} \underbrace{\begin{pmatrix} v_t \\ a_t \\ b_{\omega,t} \\ b_{a,t} \\ \omega_t \end{pmatrix}}_{x^k_t} + \begin{pmatrix} \frac{T^3}{6}I & 0 \\ 0_{4\times 3} & \frac{T}{2}\tilde{S}(q_t) \end{pmatrix} \begin{pmatrix} w_{1,t} \\ w_{2,t} \end{pmatrix}, \tag{27b}$$

$$m_{j,t+1} = m_{j,t}, \qquad j = 1, \ldots, M_t, \tag{27c}$$

where

$$\tilde{S}(q) = \begin{pmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{pmatrix}, \tag{28}$$

T denotes the sampling time, I denotes the 3×3 identity matrix and 0 denotes the 3×3 zero matrix, unless otherwise stated. The process noise w^k_t is assumed to be independent and Gaussian, with covariance $Q^k_t = \text{diag}(Q_a, Q_{b_\omega}, Q_{b_a}, Q_\omega)$.

Measurement Model – Inertial and Air Pressure Sensors

The IMU consists of accelerometers measuring accelerations $y_{a,t}$ in all three dimensions, a gyroscope measuring angular velocities $y_{\omega,t}$ in three dimensions and a magnetometer measuring the direction to the magnetic north pole. Due to the magnetic disturbances from the environment it is just the accelerometers and gyroscopes that are used for positioning. There is also a barometer available $y_{p,t}$, measuring the altitude via the air pressure. The measurements from these sensors are anti-alias filtered and down-sampled to 20 Hz. The frequency used is a tradeoff between computational load and lost accuracy. This sampling rate has shown to give good accuracy at reasonable computational load. For further details on inertial sensors, see for instance Titterton and Weston (1997), Grewal et al. (2001) and

Chatfield (1997). The measurements are related to the states according to,

$$y_{1,t} = \begin{pmatrix} y_{p,t} \\ y_{\omega,t} \\ y_{a,t} \end{pmatrix} = \underbrace{\begin{pmatrix} p_{3,t} \\ 0 \\ -R(q_t)g^e \end{pmatrix}}_{h(x_t^p)} + \underbrace{\begin{pmatrix} \multicolumn{5}{c}{0_{1\times 15}} \\ 0 & 0 & I & 0 & I \\ 0 & R(q_t) & 0 & I & 0 \end{pmatrix}}_{C(x_t^p)} \underbrace{\begin{pmatrix} v_t \\ a_t \\ b_{\omega,t} \\ b_{a,t} \\ \omega_t \end{pmatrix}}_{x_t^k} + \underbrace{\begin{pmatrix} e_{1,t} \\ e_{2,t} \\ e_{3,t} \end{pmatrix}}_{e_t},$$

which obviously is in the form (8). The measurement noise e_t is assumed Gaussian with covariance $R_t = \text{diag}(R_\omega, R_a)$.

Measurement Model – Camera

Before the camera images are used they are adjusted according to the calibration. This allows us to model the camera using the pinhole model with focal length $f = 1$, according to Ma et al. (2006) and Hartley and Zisserman (2003),

$$y_{2,t} = y_{m_j,t} = \underbrace{\frac{1}{z_t^c}\begin{pmatrix} x_t^c \\ y_t^c \end{pmatrix}}_{h^c(m_{j,t},p_t,q_t)} + e_{3,t}, \tag{29}$$

where

$$m_{j,t}^c = \begin{pmatrix} x_t^c \\ y_t^c \\ z_t^c \end{pmatrix} = R(q_t^{cb})R(q_t^{be})(m_{j,t} - p_t) + r^c, \tag{30}$$

is the position of map entry $m_{j,t}$ resolved in the camera frame. Here, r^c is a fixed vector representing the translation between the camera and the IMU (body), q_t^{cb} is the unit quaternion describing the rotation from the IMU to the camera and $R(\,\cdot\,)$ is the rotation matrix in (38). The measurement noise is distributed as

$$e_{3,t} \sim \mathsf{N}(0, R_c). \tag{31}$$

The UAV is equipped with a camera, which is synchronized in time with the inertial measurements. Images are available at 4 Hz in a resolution of 384×288 pixels. In order to use vision for feature extraction and estimation we have made use of standard camera calibration techniques, see e.g., Zhang (2000).

The features are not exactly in the form suitable for the RBPF. Hence, we are forced to use an approximation in order to obtain a practical algorithm. The standard approximation is in this case simply to linearize the camera measurement equations according to,

$$\begin{aligned} y_{m_j,t} &= h^c(m_{j,t}, p_t, q_t) + e_{3,t} \qquad (32) \\ &\approx \underbrace{h_j^c(\hat{m}_{j,t|t-1}, p_t, q_t) - H_{j,t}\hat{m}_{j,t|t-1}}_{h(x_t^p)} + H_{j,t}m_{j,t} + e_{3,t}, \quad j = 1, \ldots, M_t, \end{aligned}$$

where the Jacobian matrix $H_{j,t}$ is straightforwardly computed using the chain rule, i.e.,

$$H_{j,t} = \frac{\partial h^c}{\partial m_j} = \frac{\partial h^c}{\partial m_j^c} \frac{\partial m_j^c}{\partial m_j}. \tag{33}$$

The two partial derivatives in (33) are given by

$$\frac{\partial h^c}{\partial m_j^c} = \begin{pmatrix} \frac{1}{z^c} & 0 & -\frac{x^c}{(z^c)^2} \\ 0 & \frac{1}{z^c} & -\frac{y^c}{(z^c)^2} \end{pmatrix}, \tag{34a}$$

$$\frac{\partial m_j^c}{\partial m_j} = R(q_t^{cb})R(q_t^{be}). \tag{34b}$$

"Computing Vision Measurements"

In order to receive a camera measurement in the form (29), interest points or features have to be identified in the image. This is step 2 in Algorithm 1. In this application example, features are found using the Harris detector (Harris and Stephens, 1988), which basically extracts well-defined corners in an image. These feature points are then searched for in the subsequent images according to Algorithm 2.

Algorithm 2 Extracting and Managing Map Entries

1. Initialization. Search the entire image for features using the Harris detector. Save an 11-by-11 pixel patch around each corner.
2. Predict positions of features detected in previous images. Match saved patches in small search regions around the predicted positions using normalized cross-correlations. Also, apply a weighted criterion to the matching procedure so that a match close to the prediction is more likely.
3. Outlier rejection. If a matched feature is far from the predicted position compared to other features, the measurement is discarded.
4. In areas of the image without features, search for new features using the Harris detector. Around each detected corner an 11-by-11 pixel patch is extracted and stored.
5. Initialize the detected features in the map.

The feature detection in Algorithm 2 is performed in the 2-D image plane. However, the features found have to be initialized into the filter 3-D map. In this application, the features are known to be close to the ground and we have a good estimate of the altitude thanks to the air pressure sensor. The features are therefore initialized on the estimated ground level and adjustment are made by implicit triangulation in the particle filter. In a more general case, where the depth of the features are unknown, there are several methods available for intialization. For example, the initialization can be delayed a few time steps until the feature has been seen from several angles and its depth can be estimated by triangulation. Another alternative is to use an inverse depth parametrization for some time as in Civera et al. (2007).

This algorithm has shown to work reliably on our flight data. However, improvements can be achieved in both computational speed and detection reliability. There are more elaborate detectors available, for example the *scale-invariant feature transform* (SIFT) (Lowe, 2004), the *speeded up robust features* (SURF) (Bay et al., 2006) or the fast corner detector (Rosten and Drummond, 2006, 2005). It is also possible to refine the feature detection process even further by estimating the orientation of the patches (Davison et al., 2007). From a computer vision perspective the current environment is rather simple, hence fast and simple corner detectors can be successfully applied.

4.2 UAV Platform

The algorithm proposed has been tested using flight-test data collected from an autonomous helicopter (UAV) developed during the WITAS Project (Doherty et al., 2004). The UAV is based on a commercial Yamaha RMAX helicopter (Figure 1). The total helicopter length is 3.6 m (including main rotor), it is powered by a 21 hp two-stroke engine and it has a maximum take-off weight of 95 kg.

The avionics developed during the WITAS Project is integrated with the RMAX platform and it is based on three computers and a number of sensors. The platform developed is capable of fully autonomous flight from take-off to landing.

The sensors used for the navigation algorithm described in this paper consist of an inertial measurement unit (three accelerometers and three gyros) which provides helicopter's acceleration and angular rate along the three body axes, a barometric altitude sensor and a monocular CCD video camera. GPS position information is not used in the navigation filter described here.

The primary flight computer is a PC104 PentiumIII 700 MHz. It implements the low-level control system, which includes the control modes (take-off, hovering, path following, landing, etc), sensor data acquisition and communication with the helicopter platform. The second computer is also a PC104 PentiumIII 700 MHz, handling the image processing functionalities and controls the camera pan-tilt unit. The third computer is a PC104 Pentium-M 1.4 GHz, taking care of high-level functionalities such as path-planning, task-planning, etc.

4.3 Experiment Setup

The flight data were collected during a flight-test campaign in a training area in south of Sweden. Sensor data and on-board video were recorded during an autonomous flight. The helicopter flew autonomously a pre-planned path using a path following functionality implemented in the software architecture (Wzorek et al., 2006). The helicopter altitude was 60 m above the ground and the flight speed 3 m/s. The video camera was looking downwards and fixed to the helicopter body. The video was recorded on-board and synchronized with the sensor data. The synchronization is performed by automatically turning on a light diode when the sensor data start to be recorded. The light diode is visible in the camera image. The video is recorded on tape using an on-board video recorder

Table 1: *Available characteristics of the sensor used in the navigation algorithm.*

Sensor	Output Rate	Resolution	Bias
Accelerometers	66 Hz	1 mG	13 mG
Gyros	200 Hz	$0.1°/\mathrm{s}$	$< 0.1°/\mathrm{s}$
Barometer	40 Hz	0.1 m	-
Vision	4 Hz	384x288 pixels	-

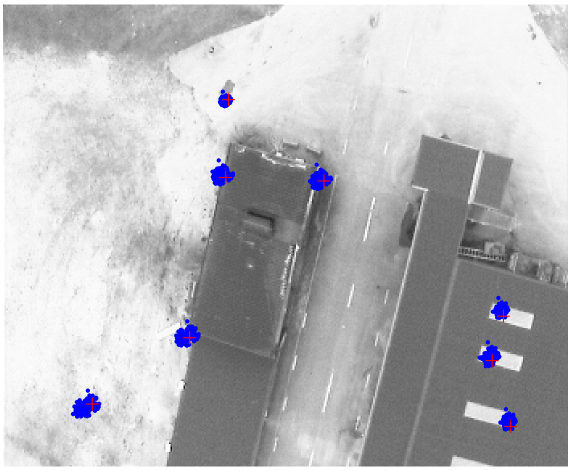


Figure 2: *The scenario seen from the on-board vision sensor, together with particle clouds representing the landmarks/map features and crosses showing the measured landmark positions.*

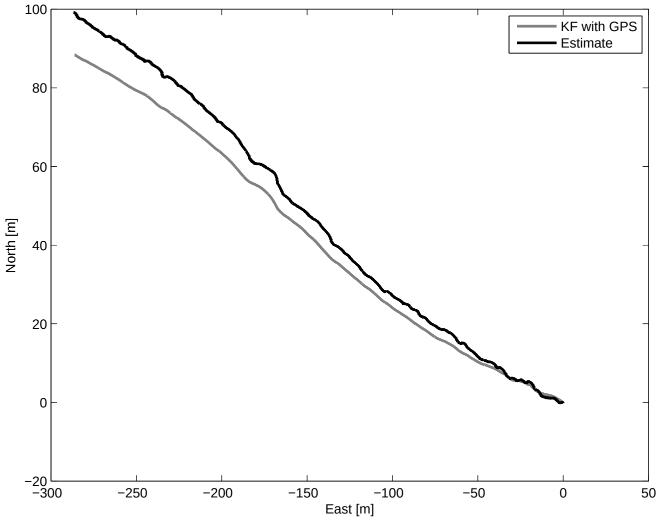
and the synchronization with the sensor data is done manually off-line. This procedure allows for synchronization accuracy of about 40 ms. The video sequence is recorded at 25 Hz frame rate. For the experiment described here the video frames were sampled at 4 Hz. The on-board sensor data are recorded at different sample rates. Table 1 provides the characteristics of the sensors used in the experiment.

4.4 Experimental Results

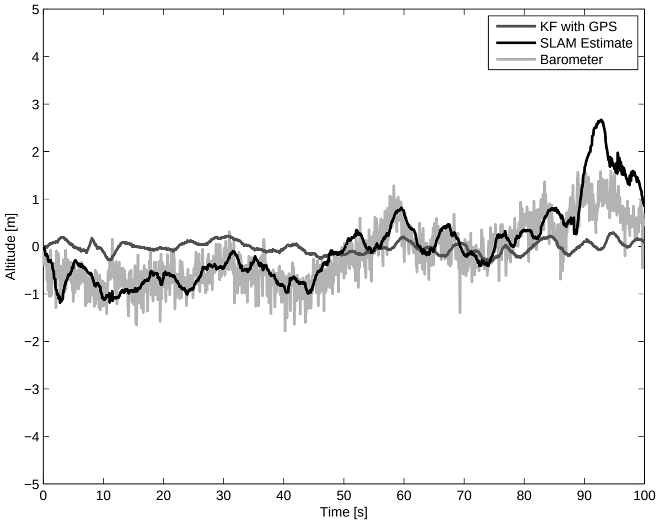
In Figure 2 the landmarks/map are depicted using the particle clouds overlayed on the current camera image.

In Figure 3(a) the Cartesian 2D position is depicted for the RBPF SLAM method, and compared against a GPS based solution. Since the SLAM method, without closing the loop, is a dead-reckoning solution it is expected to have some drift. Here, the drift is greatly reduced, compared to dead-reckoning using the inertial sensors alone. In Figure 3(b) the altitude (relative to the starting height) is depicted for the SLAM method and compared against the GPS based reference and measured air pressure.

Estimating the altitude using only vision and inertial data is problematic, since the vision measurement model will not get sufficient information in this direction. This is a well-known drawback of a monocamera solution. Hence, in order to reduce or remove a



(a) *2D position from the* SLAM *method and the* GPS *based reference.*



(b) *Altitude (relative to the starting height) for* SLAM, GPS *reference and measured with barometer.*

Figure 3: *Position and altitude of the RMAX helicopter.*

substantial drift in altitude a pressure sensor is used.

Another thing to note with the particle filter implementation of SLAM is the degeneration of the map over time. The resampling causes the map associated with the most probable features to be copied and the others to be discarded. For mapped features that have been out of sight for a while the map will then be the same for all particles after some time. This is not a problem in our example, but would be in the case of a loop-closure. The cross-correlation between the mapped features would in such an event correct even out-of-sight features. This capability would be limited here since the information about the cross-correlation among the features lies in the diversity of maps in the particles. This has been previously noted in, e.g., Bailey et al. (2006).

5 Conclusion

In this paper a PF solution to SLAM that can handle high dimensional vehicle models is presented. The solution includes an extra factorization of the filtering density compared to the standard FastSLAM algorithm. The FastSLAM algorithm factorizes the filtering distribution into two parts. Then a PF is used for filtering the states of the vehicle platform and a Kalman filter bank handles the landmarks (map). In the approach presented here, a linear Gaussian substructure in the vehicle dynamics is handled in the Kalman filter bank as well. Thus, lowering the dimension of the PF which decreases the computational load.

The UAV application consists of an RMAX helicopter, equipped with an IMU sensor (accelerometer and gyro), a pressure sensor, and a vision sensor. An on-board GPS sensor is used for evaluation, i.e., the solution presented in this paper does not rely on GPS. In an experiment the proposed sensor fusion particle filter based SLAM algorithm was successfully evaluated. Without the SLAM method the poor IMU performance would not be sufficient for navigation, whereas the SLAM technique reduces the drift introduced by the dead-reckoning sensor.

A Appendix

A.1 Coordinate Systems

Three coordinate frames are used in this paper. An earth-fixed (denoted with e), body or inertial sensor system (denoted with b), and camera system (denoted c).

The following convention is used to rotate a vector from a coordinate frame A to a coordinate frame B,

$$x_{\text{B}} = R_{\text{BA}} x_{\text{A}}. \tag{35}$$

where R_{BA} is used to denote the rotation matrix describing the rotation from A to B. Hence, the rotation from A to C, via B can be described as

$$R_{\text{CA}} = R_{\text{CB}} R_{\text{BA}}. \tag{36}$$

Another way to represent rotations is using a unit quaternion q_{A}. In this framework a vector can be rotated from A to B via

$$u_{\text{B}} = \bar{q}_{\text{A}} \odot u_{\text{A}} \odot q_{\text{A}}, \tag{37}$$

where u_{A} is the quaternion extension of the vector x_A, i.e., $u_{\text{A}} = (0 \quad x_{\text{A}}^T)^T$, and $\odot$ denotes quaternion multiplication. Furthermore, $\bar{u}$ denotes the quaternion conjugate. See e.g., Shuster (1993) and Kuipers (1999) for an introduction to unit quaternions and other rotation parameterizations.

It is straightforward to convert a given quaternion into the corresponding rotation matrix,

$$R(q) = \begin{pmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1q_2 - q_0q_3) & 2(q_1q_3 + q_0q_2) \\ 2(q_1q_2 + q_0q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2q_3 - q_0q_1) \\ 2(q_1q_3 - q_0q_2) & 2(q_2q_3 + q_0q_1) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{pmatrix}, \tag{38}$$

where $q = (q_0 \ q_1 \ q_2 \ q_3)^T$.

References

C. Andrieu and A. Doucet. Particle filtering for partially observed Gaussian state space models. *Journal of the Royal Statistical Society*, 64(4):827–836, 2002.

T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine*, 13(3):108–117, Sept. 2006.

T. Bailey, J. Nieto, and E. Nebot. Consistency of the FastSLAM algorithm. In *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, May 2006.

H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Proceedings of the ninth European Conference on Computer Vision*, May 2006.

G. Casella and C. P. Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83 (1):81–94, 1996.

A. Chatfield. *Fundamentals of High Accuracy Inertial Navigation*, volume 174. American Institute of Aeronautics and Astronautics, USA, third edition, 1997.

R. Chen and J. S. Liu. Mixture Kalman filters. *Journal of the Royal Statistical Society*, 62(3):493–508, 2000.

J. Civera, A. Davison, and J. Montiel. Inverse depth to depth conversion for monocular SLAM. In *IEEE International Conference on Robotics and Automation*, Apr. 2007.

A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings Ninth IEEE International Conference on Computer Vision*, volume 2, pages 1403–1410, Nice, France, Oct. 2003.

A. J. Davison, Y. G. Cid, and N. Kita. Real-time 3D SLAM with wide-angle vision. In *Proceedings of the 5th IFAC/EUCON Symposium on Intelligent Autonomus Vehicles*, Lisaboa, Portugal, July 2004.

A. J. Davison, I. Reid, N. Molton, and O. Strasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, June 2007.

P. Doherty, P. Haslum, F. Heintz, T. Merz, T. Persson, and B. Wingman. A distributed architecture for intelligent unmanned aerial vehicle experimentation. In *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems*, 2004.

A. Doucet, S. J. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer Verlag, New York, USA, 2001a.

A. Doucet, N. Gordon, and V. Krishnamurthy. Particle filters for state estimation of jump Markov linear systems. *IEEE Transactions on Signal Processing*, 49(3):613–624, 2001b.

H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping (SLAM): Part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110, June 2006.

E. Eade and T. Drummond. Scalable monocular SLAM. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 469–476, New York, NY, USA, June 2006.

N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings on Radar and Signal Processing*, volume 140, pages 107–113, 1993.

M. S. Grewal, L. R. Weill, and A. P. Andrews. *Global Positioning Systems, Inertial Navigation, and Integration*. John Wiley & Sons, Ltd, New York, USA, 2001.

F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation and tracking. *IEEE Transactions on Signal Processing*, 50(2):425–437, Feb. 2002.

C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, Manchester, UK, 1988.

R. Hartley and A. Zisserman. *Multiple View Geometry in computer vision*. Cambridge University Press, Cambridge, UK, 2 edition, 2003.

J. B. Kuipers. *Quaternions and Rotation Sequences*. Princeton Univerity Press, 1999.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry. *An invitation to 3-D vision – from images to geometric models*. Interdisciplinary Applied Mathematics. Springer-Verlag, 2006.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM a factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Comference on Artificial Intelligence*, Edmonton, Canada, 2002.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *In Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1151–1156, Acapulco, Mexico, 2003.

E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *Proceedings of IEEE International Conference on Computer Vision*, Oct. 2005.

E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Proceedings of European Conference on Computer Vision*, May 2006.

T. Schön, F. Gustafsson, and P.-J. Nordlund. Marginalized particle filters for mixed linear/nonlinear state-space models. *IEEE Transactions on Signal Processing*, 53(7): 2279–2289, July 2005.

T. Schön, R. Karlsson, D. Törnqvist, and F. Gustafsson. A framework for simultaneous localization and mapping utilizing model structure. In *The 10th International Conference on Information Fusion*, Quebec, Canada, Aug. 2007a.

T. Schön, D. Törnqvist, and F. Gustafsson. Fast particle filters for multi-rate sensors. In *The 15th European Signal Processing Conference (EUSIPCO 2007)*, Poznan, Poland, Sept. 2007b.

T. B. Schön, R. Karlsson, and F. Gustafsson. The marginalized particle filter in practice. In *Proceedings of IEEE Aerospace Conference*, Big Sky, MT, USA, Mar. 2006.

M. D. Shuster. A survey of attitude representations. *The Journal of Astronautical Sciences*, 41(4):439–517, 1993.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005.

D. H. Titterton and J. L. Weston. *Strapdown inertial navigation technology*. IEE radar, sonar, navigation and avionics series. Peter Peregrinus Ltd., Stevenage, UK, 1997.

M. Wzorek, G. Conte, P. Rudol, T. Merz, S. Duranti, and P. Doherty. From motion planning to control - a navigation framework for an autonomous unmanned aerial vehicle. In *21th Bristol UAV Systems Conference*, Apr. 2006.

Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, Nov. 2000.

Paper B

Unifying the Parity-Space and GLR Approach to Fault Detection with an IMU Application

Authors: David Törnqvist and Fredrik Gustafsson

Edited version of the paper: D. Törnqvist and F. Gustafsson. Unifying the parity-space and GLR approach to fault detection with an IMU application. *Automatica*, 2008. Submitted

This version of the paper has been extended with the simulation example in Section 6 from: D. Törnqvist and F. Gustafsson. Eliminating the initial state for the generalized likelihood ratio test. In *Proceedings of IFAC Symposium SAFEPROCESS*, Beijing, China, Aug. 2006.

Preliminary version: Published as Technical Report LiTH-ISY-R-2862, Dept. of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

Unifying the Parity-Space and GLR Approach to Fault Detection with an IMU Application

David Törnqvist and Fredrik Gustafsson

Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
{tornqvist, fredrik}@isy.liu.se

Abstract

Using the parity-space approach, a residual is formed by applying a projection to a batch of observed data and this is a well established approach to fault detection. Based on a stochastic state space model, the parity-space residual can be put into a stochastic framework where conventional hypothesis tests apply. In an on-line application, the batch of data corresponds to a sliding window and in this contribution we develop an improved on-line algorithm that extends the parity-space approach by taking prior information from previous observations into account. For detection of faults, the Generalized Likelihood Ratio (GLR) test is used. This framework allows for including prior information about the initial state, yielding a test statistic with a significantly higher sensitivity to faults. Another key advantage with this approach is that it can be extended to nonlinear systems using an arbitrary nonlinear filter for state estimation, and a linearized model around a nominal state trajectory in the sliding window. We demonstrate the algorithm on data from an Inertial Measurement Unit (IMU), where small and incipient magnetic disturbances are detected using a nonlinear system model.

1 Introduction

The parity-space method is a widely used method to find a residual for fault detection. It was first described in Chow and Willsky (1984), but is described and used in many publications for example Basseville and Nikiforov (1993), Ding et al. (1999), Gertler (1997, 1998). The parity-space approach is a well established approach to fault detection, and one of its main features is the intuitive geometrical interpretation of the residual that is obtained by projecting a batch of observations onto the parity space. The projection assures that the influence of the unknown initial state in the data batch as well as external disturbances do not affect the residual from which diagnosis decisions are made.

The classical parity-space approach applies to linear deterministic models, and has been extended to a statistical framework (Gustafsson, 2002, 2007, Fouladirad and Nikiforov, 2005), which makes it applicable to stochastic linear state-space models with additive faults. The performance limits for various noise distributions have been studied in Hendeby and Gustafsson (2006). Its main limitations are:

- It may have a poor sensitivity to faults, in particular for short data batches.
- It cannot handle prior knowledge about the initial state. For instance, a state estimate may be available from observations outside the sliding window.
- It is not easily extended to nonlinear systems.

It is the purpose of this contribution to extend the stochastic parity-space approach to handle these drawbacks.

The main idea is to incorporate prior information of the state when forming the residual. In this way, the sensitivity to faults increases automatically. This increased sensitivity can be used to keep the batch window short while keeping a sufficient sensitivity to faults and in this way decrease the delay until detection. The final result is an algorithm which is related to one of the methods suggested in Willsky and Jones (1976). This seminal paper applies the Generalized Likelihood Ratio (GLR) test to the same stochastic state-space model, and to limit the computational complexity, the authors propose an algorithm where faults are restricted to a sliding time window. The final algorithm consists of two filters from which all relevant test statistics can be computed.

Further, with a good prior of the initial state and a short batch window, nonlinear systems can be linearized along a nominal trajectory and the parity-space residual is approximated using the linearized model. The shorter window, the better approximation. This means that a nonlinear filter, as the Extended Kalman Filter (EKF), the Unscented Kalman Filter (UKF) or the Particle Filter (PF), can be run on-line with a time lag corresponding to the batch window, and the residual is continuously monitored on this batch window. If a fault is detected, the filter can be put in simulation mode. The ability to handle nonlinear models in a natural way is a clear improvement to the GLR test in Willsky and Jones (1976).

To demonstrate the algorithm, an application of estimating orientation based on measurements from an Inertial Measurement Unit (IMU) is studied. The unit consists of accelerometers, gyroscopes and magnetometers. This is a classical filtering problem, where the state-space model basically says that the orientation is the integral of measured gyroscope values. This model is subject to a drift which evolves linearly in time, and needs external support. Here, the magnetic field sensors normally functions as a compass and provides a measurement with two degrees of freedom, and the accelerometer can, if there is no external acceleration, be used as an inclinometer providing a measurement relation with three degrees of freedom.

The outline is as follows. Section 2 defines the models used. Using these models, various ways of estimating the initial state in a time window is described in Section 3. The parity-space approach and its relation to different initial state estimates is then described in Section 4. Using the parity-space residuals formed in Section 4, the GLR test in Section 5 can be used for detection. A simulation example in Section 6 shows the detection performance and the framework is applied to detect magnetic disturbances in Section 7. Finally, conclusions are provided in Section 8.

2 Modeling

This section describes the relation between state-space modeling and modeling over a sliding window or batch. An efficient fault parameterization is also introduced.

2.1 System Model

Let us consider a discrete-time dynamical system in the form

$$x_{t+1} = F_t x_t + G_t^u u_t + G_t^f f_t + G_t^v v_t, \tag{1a}$$

$$y_t = H_t x_t + H_t^u u_t + H_t^f f_t + e_t, \tag{1b}$$

where x_t is the state vector, u_t an input signal, f_t a scalar time-varying fault profile, y_t a measurement, v_t process noise and e_t the measurement noise. The fault detection test can here be stated as testing if $f_t = 0$ or $f_t \neq 0$.

2.2 Batch Model

The batch form is often used in fault detection and diagnosis Chow and Willsky (1984), Gertler (1998), Gustafsson (2001). Stack L signal values to define the batched signal vectors like $\mathbb{Y} = \left(y_{t-L+1}^T, \ldots, y_t^T\right)^T$, for all signals. Then, the outputs in that window will be given by

$$\mathbb{Y} = \mathcal{O}_t x_{t-L+1} + \bar{H}_t^u \mathbb{U} + \bar{H}_t^f \mathbb{F} + \bar{H}_t^v \mathbb{V} + \mathbb{E}, \tag{2}$$

with the extended observability matrix

$$\mathcal{O}_t = \begin{pmatrix} H_{t-L+1} \\ H_{t-L+2} F_{t-L+1} \\ \vdots \\ H_t \prod_{k=t-1}^{t-L+1} F_k \end{pmatrix}. \tag{3}$$

The matrices determining how the remaining signals affect the system are described by

$$\bar{H}_t^s = \begin{pmatrix} H_{t-L+1}^s & 0 & \cdots & 0 \\ H_{t-L+2} G_{t-L+1}^s & H_{t-L+2}^s & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ H_t \prod_{k=t-1}^{t-L+2} F_k G_{t-L+1}^s & \cdots & H_t G_{t-1}^s & H_t^s \end{pmatrix}, \tag{4}$$

where $s = \{u, f, v\}$. For simplicity, denote $\mathbb{Z} = \mathbb{Y} - \bar{H}_t^u \mathbb{U}$, implying that the system (2) can be written as

$$\mathbb{Z} = \mathcal{O}_t x_{t-L+1} + \bar{H}_t^f \mathbb{F} + \bar{H}_t^v \mathbb{V} + \mathbb{E}. \tag{5}$$

The system is assumed to be observable, that is, $\mathcal{O}_t$ has full column rank and all states will have a unique influence in the output. However, it is often possible to observe a subspace of the state space and the system can therefore be partially observable. In order to handle such a system, the observable subspace has to be determined. It is given by the singular value decomposition

$$\mathcal{O}_t = U_1 \Sigma V_1^T, \tag{6}$$

where U_1 spans the observable subspace. The state $\bar{x} \triangleq \Sigma V_1^T x$ $(\dim(\bar{x}) < \dim(x))$ is now an observable state. The system (2) can then be rewritten as

$$\mathbb{Y} = U_1 \bar{x} + \bar{H}_t^u \mathbb{U} + \bar{H}_t^f \mathbb{F} + \bar{H}_t^v \mathbb{V} + \mathbb{E}. \tag{7}$$

Now, all methods described in this article can be used with $\mathcal{O}_t$ exchanged for U_1. When a prior estimate of the original estimate is available, as in Section 3.4, the state estimate and its covariance have to be transformed as

$$\hat{\bar{x}}^{(1)} = \Sigma V_1^T \hat{x}^{(1)} \tag{8}$$

$$P_{\hat{\bar{x}}^{(1)}}^{(1)} = \Sigma V_1^T P^{(1)} V_1 \Sigma. \tag{9}$$

2.3 Fault Model

It is convenient for the analysis to assume that the fault is constant over time, while in practice a fault is often increasing or decreasing slowly in magnitude over time. To include such incipient faults, the fault profile f_t can be modeled with a low dimensional parametrization as

$$\mathbb{F} = \Phi^T \theta_t \Rightarrow \bar{H}^\theta \triangleq \bar{H}^f \Phi^T. \tag{10}$$

For the batch model, this model implies that the fault parameter can be considered constant over the data window. Thus, a scalar time-varying parameter is replaced with a constant vector. This facilitates the analysis significantly.

In this paper, the Chebyshev polynomial is used as basis functions, see Abramowitz and Stegun (1965), Rivlin (1974). An example of the first three basis functions which are orthogonal Chebyshev polynomials over a window of 10 samples is given in Figure 1(a) and a parameterized disturbance from a magnetometer is shown in Figure 1(b).

2.4 Model Summary

Combining the batch data model with the fault model, the total model that will be studied in the sequel can be summarized as

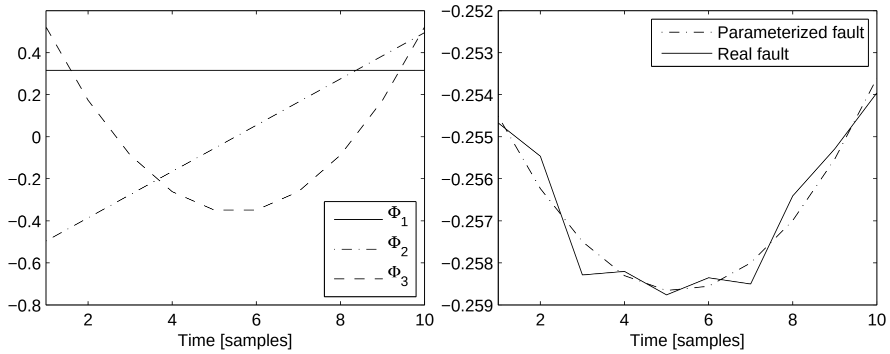
$$\mathbb{Z} = \mathcal{O}_t x_{t-L+1} + \bar{H}_t^\theta \theta_t + \mathbb{N}_t, \tag{11}$$

where

$$\mathbb{N}_t \triangleq \bar{H}_t^v \mathbb{V} + \mathbb{E}, \tag{12}$$

and

$$S \triangleq \mathsf{Cov}(\mathbb{N}_t). \tag{13}$$



***(a)** Fault basis* ***(b)** Parameterized fault*

***Figure 1:** The basis functions (a) are used to parameterize the fault from a magnetic disturbance (b).*

3 State Estimation

There are two unknowns in the data model (11), the initial state x_{t-L+1} and the fault profile parameter vector θ_t. This section describes different ways to estimate the initial state that are instrumental for the results in the next section. Time indices are here omitted for simplicity, e.g., $x = x_{t-L+1}$.

3.1 Least Squares Estimation

The least squares estimate of the initial state is given by

$$\hat{x} = \arg\min_{x} \|\mathcal{O}x - \mathbb{Z}\|_2^2, \tag{14}$$

which gives the solution

$$\hat{x} = \mathcal{O}^{\dagger}\mathbb{Z}. \tag{15}$$

Here, $\mathcal{O}^{\dagger}$ denotes the Moore-Penrose pseudo inverse of $\mathcal{O}$ (see Golub and van Loan (1996)). This estimate does not account for the noise covariance S, and is included here because it is closely related to the original parity-space approach.

This estimate is unbiased in case of no fault, since

$$\mathsf{E}\,\hat{x} = \mathsf{E}\,\mathcal{O}^{\dagger}\mathbb{Z} = \mathsf{E}\left(x + \mathcal{O}^{\dagger}\mathbb{N}\right) = \mathsf{E}\,x, \tag{16}$$

and

$$\mathsf{Cov}(\hat{x}) = \mathcal{O}^{\dagger}S\mathcal{O}^{\dagger T}. \tag{17}$$

Observe that $\mathcal{O}^{\dagger}\mathcal{O} = I$ since the system is assumed to be observable and hence $\mathcal{O}$ has full column rank. In presence of faults, the expected value would be

$$\mathsf{E}\,\hat{x} = \mathsf{E}\,\mathcal{O}^{\dagger}\mathbb{Z} = x + \mathcal{O}^{\dagger}\bar{H}^{\theta}\theta. \tag{18}$$

3.2 Minimum Variance State Estimation

The equation system $\mathcal{O}x + \mathbb{N} = \mathbb{Z}$ is known as the *general Gauss-Markov linear model*. If $S \succ 0$, the parameters x can be estimated with minimum variance by solving the generalized least squares problem

$$\hat{x} = \arg\min_{x} (\mathcal{O}x - \mathbb{Z})^T S^{-1} (\mathcal{O}x - \mathbb{Z}), \tag{19}$$

see Björck (1996). This can be rewritten as the least squares problem

$$\hat{x} = \arg\min_{x} \|S^{-1/2}\mathcal{O}x - S^{-1/2}\mathbb{Z}\|_2^2, \tag{20}$$

which has the solution

$$\hat{x} = (S^{-1/2}\mathcal{O})^{\dagger} S^{-1/2}\mathbb{Z}. \tag{21}$$

This estimate is distributed according to

$$\hat{x} \sim \mathsf{N}\Big(x + (S^{-1/2}\mathcal{O})^{\dagger} S^{-1/2}\bar{H}^{\theta}\theta, (\mathcal{O}^T S^{-1}\mathcal{O})^{-1}\Big). \tag{22}$$

3.3 Filtered Estimate from Past Data

Given that the data before the sliding time window are not affected by a fault (the fault detection test assumes a fault in the batch window), a suitable filter can be applied to the original model (1). For a nonlinear system a nonlinear filter, e.g., EKF, UKF, or PF, can be applied. This yields another estimate of the initial state in the batch window, independent of (21). The distribution for the initial state can then be approximated by a Gaussian distribution

$$\hat{x} \sim \mathsf{N}(x, P), \tag{23}$$

to facilitate the linear framework in the parity-space methods.

3.4 Smoothed State Estimate

Given the two estimates (22) and (23) based on independent data sets, a combined estimate can be formed similar to smoothing. Combining two independent estimates is often referred to as the *sensor fusion problem*, see for instance Gustafsson (2001).

Denote the two distributions in (22) and (23)

$$\hat{x}^{(i)} \sim \mathsf{N}(x, P^{(i)}), \quad i = 1, 2. \tag{24}$$

Then, the smoothed initial state is estimated by

$$\hat{x} = P\Big({P^{(1)}}^{-1}\hat{x}^{(1)} + {P^{(2)}}^{-1}\hat{x}^{(2)}\Big), \tag{25}$$

where

$$P \triangleq \Big({P^{(1)}}^{-1} + {P^{(2)}}^{-1}\Big)^{-1}. \tag{26}$$

The expected value of this estimate is derived using (22) and (24)

$$\mathsf{E}\,\hat{x} = \underbrace{P({P^{(1)}}^{-1} + {P^{(2)}}^{-1})}_{=I}x + P{P^{(2)}}^{-1}(S^{-1/2}\mathcal{O})^{\dagger}S^{-1/2}\bar{H}^{\theta}\theta$$

$$= x + P\mathcal{O}^{T}S^{-1}\bar{H}^{\theta}\theta \quad (27)$$

Note that $\mathsf{Cov}(\hat{x}) = P \preceq P^{(2)}$ which means that the covariance of the estimate is decreased when prior information is used.

4 Fault Detection with Parity Space

The classical parity space uses a projection to construct a residual insensitive to the initial state. This section shows that this is equivalent to estimate the initial state by least squares from data in the batch and substract the estimated influence. The parity-space framework is also extended for the use of prior information about the initial state.

4.1 Classical Parity Space

The parity-space idea is to project the output on a space that is orthogonal to the influence of the initial state, the parity space. The space influenced by the initial state is spanned by $\mathcal{O}$, see (5). Thus, define $\mathcal{B}_{\mathcal{O}^{\perp}}$ as an orthonormal basis for $\mathcal{R}(\mathcal{O}^{\perp})$. Now, a residual insensitive to the initial state can be defined as

$$r = \mathcal{B}_{\mathcal{O}^{\perp}}^{T}\,\mathbb{Z}. \quad (28)$$

The batch model (11) gives an expression for the residual suitable for analysis as

$$r = \mathcal{B}_{\mathcal{O}^{\perp}}^{T}\,\bar{H}^{\theta}\theta + \mathcal{B}_{\mathcal{O}^{\perp}}^{T}\,\mathbb{N} \sim \mathsf{N}(\mathcal{B}_{\mathcal{O}^{\perp}}^{T}\,\bar{H}^{\theta}\theta, \mathcal{B}_{\mathcal{O}^{\perp}}^{T}\,S\,\mathcal{B}_{\mathcal{O}^{\perp}}). \quad (29)$$

The normalized residual is then

$$\bar{r} = \underbrace{(\mathcal{B}_{\mathcal{O}^{\perp}}^{T}\,S\,\mathcal{B}_{\mathcal{O}^{\perp}})^{-1/2}\,\mathcal{B}_{\mathcal{O}^{\perp}}^{T}}_{\triangleq\bar{W}_1^{T}}\,\mathbb{Z} \sim \mathsf{N}(\bar{W}_1^{T}\bar{H}^{\theta}\theta, I). \quad (30)$$

4.2 Parity Space as Least Squares State Estimation

By inserting the least squares estimate (15) into the batch model (11), the model error is obtained as

$$\varepsilon = \mathbb{Z} - \mathcal{O}\hat{x} = \mathbb{Z} - \underbrace{\mathcal{O}\mathcal{O}^{\dagger}}_{\mathcal{P}_{\mathcal{O}}}\mathbb{Z} = (I - \mathcal{P}_{\mathcal{O}})\mathbb{Z} = \mathcal{P}_{\mathcal{O}^{\perp}}\mathbb{Z}. \quad (31)$$

Note that $\mathcal{P}_{\mathcal{O}} = \mathcal{O}\mathcal{O}^{\dagger}$ is an orthogonal projection onto the space spanned by $\mathcal{O}$ and $\mathcal{P}_{\mathcal{O}^{\perp}} = I - \mathcal{P}_{\mathcal{O}}$ is the orthogonal projection onto the complementary to $\mathcal{O}$, denoted $\mathcal{O}^{\perp}$ Meyer (2000). The projector can also be written as

$$\mathcal{P}_{\mathcal{O}^{\perp}} = \mathcal{B}_{\mathcal{O}^{\perp}}\,\mathcal{B}_{\mathcal{O}^{\perp}}^{T}, \quad (32)$$

where $\mathcal{B}_{\mathcal{O}^\perp}$ is an orthonormal basis for $\mathcal{R}(\mathcal{O}^\perp)$.

Since the model error is computed by a rank deficient projection, the information in the prediction error can be represented by a vector of lower dimension. This low-dimensional vector is here denoted the residual. Note that $\mathcal{B}_{\mathcal{O}^\perp}^T \mathcal{B}_{\mathcal{O}^\perp} = I$ due to orthonormality and that the dimension is the rank of the projection in (32). The residual can be written as

$$r = \mathcal{B}_{\mathcal{O}^\perp}^T \varepsilon = \mathcal{B}_{\mathcal{O}^\perp}^T \mathcal{B}_{\mathcal{O}^\perp} \mathcal{B}_{\mathcal{O}^\perp}^T \mathbb{Z} = \mathcal{B}_{\mathcal{O}^\perp}^T \mathbb{Z}. \tag{33}$$

This expression equals the residual created by the classical parity-space method in Section 4.1. That is, we have shown that a hypothesis test on the model error can be reduced to a hypothesis test on the residual from the parity-space approach. Thus, these two approaches coincide.

4.3 Parity Space with Smoothing

To construct the test statistic, it is desirable to have the model error ε as a function of the measurements, $\mathbb{Z}$, and the state estimate from the previous data, $\hat{x}^{(1)}$. Thus,

$$\begin{aligned}\varepsilon = \mathbb{Z} - \mathcal{O}\hat{x} &= \mathbb{Z} - \mathcal{O}P\Big(P^{(1)^{-1}}\hat{x}^{(1)} + P^{(2)^{-1}}\hat{x}^{(2)}\Big)\\ &= \underbrace{\Big(I - \mathcal{O}PP^{(2)^{-1}}(S^{-1/2}\mathcal{O})^\dagger S^{-1/2}\Big)}_{\triangleq W_2^T}\mathbb{Z} - \mathcal{O}PP^{(1)^{-1}}\hat{x}^{(1)}\\ &= W_2^T\mathbb{Z} - \mathcal{O}PP^{(1)^{-1}}\hat{x}^{(1)}\end{aligned} \tag{34}$$

and the covariance is

$$\mathsf{Cov}(\varepsilon) = W_2^T S W_2 + \mathcal{O}\underbrace{PP^{(1)^{-1}}P}_{\triangleq Q}\mathcal{O}^T = \begin{pmatrix} W_2^T & \mathcal{O}\end{pmatrix}\begin{pmatrix} S & 0\\ 0 & Q\end{pmatrix}\begin{pmatrix} W_2\\ \mathcal{O}^T\end{pmatrix}. \tag{35}$$

It is important to note that the matrix W_2^T is time varying due to the dependence of the covariance from the filter used on data before the window. However, if a Kalman filter is used it will become stationary and then the covariance converges to a matrix which can be computed by solving the Riccati equation.

Lemma 1
Assume that S and Q are both positive definite matrices, then the matrix in (35) *is positive definite.*

Proof: See Appendix A.1. □

Since $Q = PP^{(1)^{-1}}P$ where $P^{(1)}$ and P are positive definite matrices, Q is also positive definite. Then, according to Lemma 1, the covariance of ε is positive definite and thus invertible. The normalized residual can therefore be expressed as

$$\bar{r} = \mathsf{Cov}(\varepsilon)^{-1/2}\varepsilon = \underbrace{\mathsf{Cov}(\varepsilon)^{-1/2}W_2^T}_{\triangleq \bar{W}_2^T}\mathbb{Z} - \mathsf{Cov}(\varepsilon)^{-1/2}\mathcal{O}PP^{(1)^{-1}}\hat{x}^{(1)}. \tag{36}$$

5 GLR Test

Detecting faults is done using the residuals derived in Section 4. Using the information about the noise distribution of the residuals, a GLR test statistic is formed. To make the decision if a fault is present or not, the test statistic is compared to a threshold from the chi-square distribution.

5.1 Test Statistic

Fault detection is here considered as detecting whether the fault is zero or not. This approach corresponds to the hypothesis test

$$\mathcal{H}_0 :\ \theta = 0 \tag{37a}$$

$$\mathcal{H}_1 :\ \theta \neq 0. \tag{37b}$$

Using the normalized residuals (30) or (36), the hypothesis test can be written as

$$\mathcal{H}_0 :\ \bar{r} \sim \mathsf{N}(0, I) \tag{38a}$$

$$\mathcal{H}_1 :\ \bar{r} \sim \mathsf{N}(\bar{W}_i^T \bar{H}^\theta \theta, I). \tag{38b}$$

where $i \in \{1, 2\}$ (depending on method). This yields the log-likelihood ratio

$$\begin{aligned} L = \sup_\theta 2 \log \left(\frac{p(\bar{r}|\,\mathcal{H}_1)}{p(\bar{r}|\,\mathcal{H}_0)} \right) &= \sup_\theta 2 \log \frac{e^{-\frac{1}{2}\|\bar{r} - \bar{W}_i^T \bar{H}^\theta \theta\|_2^2}}{e^{-\frac{1}{2}\|\bar{r}\|_2^2}} \\ &= \sup_\theta -\Big(\|\bar{r} - \bar{W}_i^T \bar{H}^\theta \theta\|_2^2 - \|\bar{r}\|_2^2 \Big), \end{aligned} \tag{39}$$

which is maximized for $\theta = (\bar{W}_i^T \bar{H}^\theta)^\dagger \bar{r}$. Then

$$L = -\Big(\|(I - \underbrace{\bar{W}_i^T \bar{H}^\theta (\bar{W}_i^T \bar{H}^\theta)^\dagger}_{\mathcal{P}_{\bar{W}_i^T \bar{H}^\theta}})\bar{r}\|_2^2 - \|\bar{r}\|_2^2 \Big) = \bar{r}^T \mathcal{P}_{\bar{W}_i^T \bar{H}^\theta} \bar{r} \tag{40}$$

5.2 Statistics

To choose suitable thresholds for the test statistics above, it is necessary to compute their distributions. While having Gaussian noise, the test statistics will be chi-square distributed variables, see Kay (1998). The normalized residual is distributed as

$$\bar{r} \sim \mathsf{N}(\bar{W}_i^T \bar{H}^\theta \theta, I), \tag{41}$$

where $\theta = 0$ under the null hypothesis (38a). The test statistic is then distributed as the non-central chi-square distribution

$$L = \bar{r}^T \mathcal{P}_{W_i^T \bar{H}^\theta} \bar{r} \sim \chi_\nu'^2(\lambda), \tag{42}$$

where $\nu = \operatorname{rank}(W_i^T \bar{H}^\theta)$ and

$$\lambda = (W_i^T \bar{H}^\theta \theta)^T \mathcal{P}_{W_i^T \bar{H}^\theta} W_i^T \bar{H}^\theta \theta = (W_i^T \bar{H}^\theta \theta)^T W_i^T \bar{H}^\theta \theta. \tag{43}$$

Observe that $\lambda = 0$ in the fault-free case and then the test statistic is distributed according to the central chi-square distribution $L \sim \chi^2_\nu$. The threshold is then chosen from the chi-square distribution so that the fault-free hypothesis is rejected erroneously only with a small probability.

6 Simulation Example

To show the performance of the fault detection algorithms, a DC-motor has been simulated. The DC-motor can have a fault which is interpreted as a torque disturbance or a fault in the voltage supply.

6.1 Modeling

The continuous-time transfer function from input voltage to axle angle is

$$G(s) = \frac{1}{s(s+1)}. \tag{44}$$

Using the state variables x_1 (angle) and x_2 (angular velocity), a sampled version of the DC-motor can be written as

$$x_{t+1} = Fx_t + G^u u_t + G^f f_t + G^w w_t \tag{45a}$$

$$y_t = Hx_t + e_t, \tag{45b}$$

with the system matrices

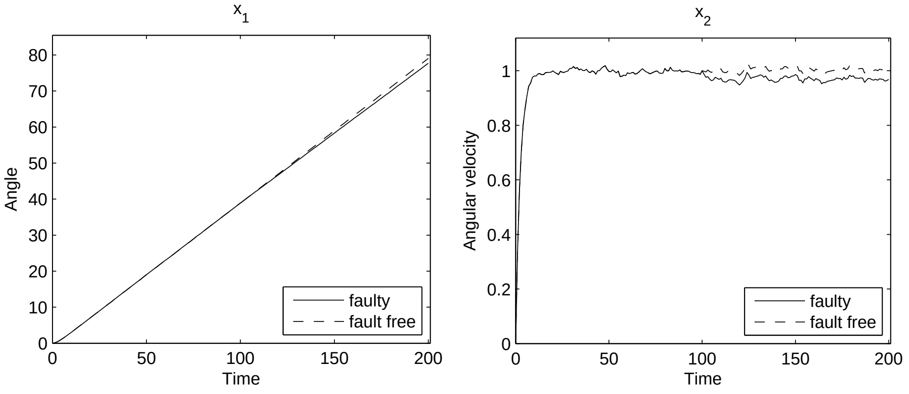
$$F = \begin{pmatrix} 1 & 1-e^{-T} \\ 0 & e^{-T} \end{pmatrix}, \qquad G^u = \begin{pmatrix} T-(1-e^{-T}) \\ 1-e^{-T} \end{pmatrix},$$

$$G^w = G^f = G^u, \qquad H = \begin{pmatrix} 1 & 0 \end{pmatrix},$$

where $w_t \sim \mathsf{N}\left(0, \left(\frac{\pi}{180}\right)^2\right)$, $e_t \sim \mathsf{N}\left(0, \left(\frac{\pi}{180}\right)^2\right)$ and the sample interval used is $T = 0.4\,\text{s}$.

6.2 Simulations

In the simulation, the system has been simulated for 200 samples in time with the unit step as input. After 100 fault-free samples, a constant fault is introduced with the magnitude of $\frac{2\pi}{180}$, which is present until the last sample. The fault can be seen as a torque disturbance which causes a drop in the angular velocity, x_2, and a slope change in the angle, x_1. With this setup, 2000 Monte Carlo (MC) simulations have been carried out. One instance of the state trajectory realizations is shown in Figure 2 where also the fault-free trajectory is shown using the same noise realization. The drop in angular velocity when the fault is present can be seen in Figure 2(b). The angle, which is measured, has a small change in the slope which is hardly noticeable in Figure 2(a). The fault detection method using smoothing, includes a time-varying term dependent on the estimation covariance from the Kalman filter. After a transient period the Kalman filter will become stationary which



(a) *Angle* ***(b)*** *Angular velocity*

Figure 2: *Plots of the state trajectories with and without fault influence for the* DC-*motor in* (45).

means that the estimation covariance converges. Figure 3 shows $\sqrt{\operatorname{tr} P^{(1)}}$, a measure of the state covariance, $P^{(1)}$. Note that $P^{(1)}$ converges already after approximately 15 samples and the smoothing method can then be considered time invariant.

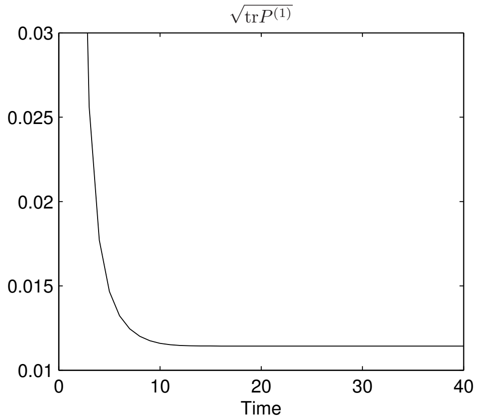


Figure 3: $\sqrt{\operatorname{tr} P^{(1)}}$ *is used as a measure of the state estimation covariance.*

6.3 Fault Detection

Fault detection on the simulated data has been made both with and without making use of the known fault structure. Since the fault is a step disturbance, a basis of dimension one can be used to describe its behavior. For fault detection a sliding window of 8 samples is used and for the smoothed method the Kalman filter estimate from previous data is used as well. For the discussions below, some definitions have to be made. The alarm rate is computed as

$$a_r(t) = \frac{\text{\# L(t)} > \gamma}{\text{\# MC-simulations}}, \tag{46}$$

where $L(t)$ is the test statistic and γ the threshold for detection. The probability of detection, P_D, is defined as the alarm rate during presence of fault. Probability of false alarm, P_{FA}, is defined as the alarm rate during fault-free periods.

No Knowledge about Fault Structure (Case A)

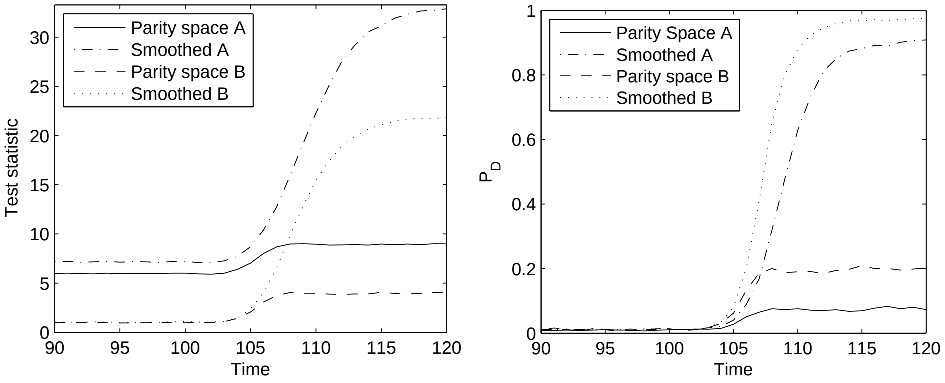
First, the fault detection methods are used without assuming knowledge about the structure of the fault. The results can be seen in Figures 4(a)–4(c). The average over the MC-simulations of the test statistics (40) is shown in Figure 4(a). Note that the difference in average during the fault-free part between the methods is due to different degrees of freedom of the test statistic. The parity-space method has 6 degrees of freedom whereas the smoothed method has 7. The alarm rate versus time is shown in Figure 4(b). The false alarm rate has been set to 1% and it and the actual alarm rate during the fault-free parts of the simulation corresponds to this value. The method with smoothing outperforms the parity-space approach, but the comparison is unfair since the estimation is based on a different amount of data. In Figure 4(c), the Receiver Operation Characteristics (ROC) is plotted. The ROC-curve shows how the probability of detection for a fault varies versus the probability of false alarm. Guessing corresponds to a the straight line between $P_{FA} = P_D$. In this case the values for P_D is estimated one window after the entrance of the fault. This is done since the Kalman filter estimate, used in the smoothed method, will be influenced by the fault afterward. More about ROC-curves can be found in Kay (1998).

Using the Structure of the Faults (Case B)

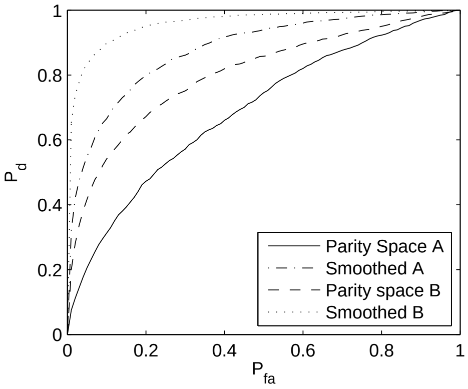
Secondly, the structure of the fault is used to enhance the fault detectability. Since it is here assumed that the fault is a step, a one dimensional basis function is used to describe the fault. It is discussed in Section 5.2 that the average of the test statistics should equal the degrees of freedom for the test statistic. Due to the parameterization, the degree of freedom is 1 and the effect of this can be seen in Figure 4(a) where the averages of the test statistics (40) are 1 in the fault-free part. The lower average during the fault-free part makes the relative increase to the faulty part larger and the probability of detection is therefore higher than without structure. Figure 4(b) shows that the probability of detection is considerably increased for both methods due to the use of the fault structure. The ROC-curves in Figure 4(c) have therefore also shifted toward higher detection rates.

7 Application Example: Detection of Magnetic Disturbances

In many situations, the orientation of a system is of interest. To estimate the orientation, an IMU can be used. The IMU normally contains three dimensional accelerometers and gyroscopes. However, the measurements in this example are made with an IMU from



***(a)** Test statistic* ***(b)** Alarm rate*



***(c)** ROC-curve*

Figure 4: *The averages of the test statistics* (40) *are given in (a), the alarm rate in (b) and a plot of the* ROC *curve in (c). A and B denote with and without using the structure of the fault respectively.*

Xsens Technologies, see (Xsens Technologies B.V.), which is augmented with a magnetometer (compass) to enhance the orientation estimation. The magnetometer measures the weak magnetic field of the earth and is therefore very sensitive to disturbances from ferro-magnetic objects or electric fields. The purpose of this example is to show how these disturbances can be detected.

7.1 Modeling

To describe the behavior of the IMU, two coordinate systems are used, one fixed to the world and one fixed in the IMU (body). Denote the two coordinate systems with w and b respectively. The quaternion, q_t, is used to describe the rotation between the two coordinate systems. For a thorough review of quaternions, see Kuipers (1999). The following quaternion dynamics was derived in Törnqvist (2006)

$$q_{t+1} = \underbrace{\left(I_4 + \frac{T}{2} S(\omega^b_{bw,t})\right)}_{\triangleq F_t} q_t + \underbrace{\frac{T}{2} S'(q_t)}_{\triangleq G^v_t} v_t, \tag{47}$$

where

$$S(\omega) = \begin{pmatrix} 0 & -\omega_x & -\omega_y & -\omega_z \\ \omega_x & 0 & \omega_z & -\omega_y \\ \omega_y & -\omega_z & 0 & \omega_x \\ \omega_z & \omega_y & -\omega_x & 0 \end{pmatrix}, \quad S'(q_t) = \begin{pmatrix} -q_1 & -q_2 & -q_3 \\ q_0 & -q_3 & q_2 \\ q_3 & q_0 & -q_1 \\ -q_2 & q_1 & q_0 \end{pmatrix}.$$

The sensors that are available in the system are gyroscopes, accelerometers and magnetometers (compass). The measurements from the gyroscopes are, as seen in (47), directly incorporated in the dynamics and are therefore not described in the measurement equations.

The accelerometers measures the earth gravitational field described in the b-system. For this example, it is assumed that the sensor is not accelerated or that the acceleration is known. The measurement equation can therefore be written as

$$y_{a,t} = g^b + e_{a,t} = \underbrace{R(q_{bw}) g^w}_{\triangleq h_a(q_{bw})} + e_{a,t}, \tag{48}$$

where the rotation matrix

$$R(q) = \begin{pmatrix} (q_0^2 + q_1^2 - q_2^2 - q_3^2) & 2(q_1 q_2 + q_0 q_3) & 2(q_1 q_3 - q_0 q_2) \\ 2(q_1 q_2 - q_0 q_3) & (q_0^2 - q_1^2 + q_2^2 - q_3^2) & 2(q_2 q_3 + q_0 q_1) \\ 2(q_1 q_3 + q_0 q_2) & 2(q_2 q_3 - q_0 q_1) & (q_0^2 - q_1^2 - q_2^2 + q_3^2) \end{pmatrix}, \tag{49}$$

g is the gravity vector and e_a is the measurement noise.

The reading from the magnetometer is a normalized vector, $\bar{n}^b_{np}$, given in the b-system and pointing along the earth magnetic field. Since the earth magnetic field is very weak, it can easily be disturbed by an electric motor or ferromagnetic objects such as beams or other iron constructions. Such a disturbance can be represented by an unknown vector, here denoted $d_{m,t}$. The measurement equation can thus be written as

$$y_{m,t} = \bar{n}^b_{np,t} + d_{m,t} + e_{m,t} = \underbrace{R(q_{bw,t})\bar{n}^w_{np}}_{\triangleq h_m(q_{bw})} + d_{m,t} + e_{m,t}, \tag{50}$$

where $R(\star)$ is the rotation matrix given in (49), $\bar{n}_{np}$ is the normalized earth magnetic field vector and $e_{m,t}$ is the measurement noise.

The total measurement equation becomes

$$y_t = \begin{pmatrix} y_{a,t} \\ y_{m,t} \end{pmatrix} = h(q_{bw,t}) + \underbrace{\begin{pmatrix} 0 \\ I \end{pmatrix}}_{\triangleq H_{d,t}} d_{m,t} + e_t. \tag{51}$$

Linearization

To estimate the orientation using the EKF and to use linear theory for fault detection, the system must be linearized. The dynamic equation (47) is already linear and time-varying, but the measurement equation (51) is nonlinear. It is therefore linearized with a first order Taylor series

$$y_t = h(q_t) + e_t \approx h(\hat{q}_{t|t-1}) + H_t(q_t - \hat{q}_{t|t-1}) + H_{d,t} d_{m,t} + e_t, \tag{52}$$

where:

$$H_t = \left. \frac{\partial h(q)}{\partial q} \right|_{q=\hat{q}_{t|t-1}} \tag{53}$$

To form the linearized system, a new measurement variable is computed with known information as

$$\tilde{y}_t \triangleq y_t - h(\hat{q}_{t|t-1}) + H_t \hat{q}_{t|t-1} = H_t q_t + H_{d,t} d_{m,t} + e_t. \tag{54}$$

Then the linearized system can now be written as

$$q_{t+1} = F_t q_t + G^v_t v_t \tag{55a}$$

$$\tilde{y}_t = \underbrace{\begin{pmatrix} H^a_t \\ H^m_t \end{pmatrix}}_{=H_t} q_t + \underbrace{\begin{pmatrix} 0 \\ I \end{pmatrix}}_{=H^d_t} d_{m,t} + e_t. \tag{55b}$$

The batched form discussed in Section 2.2 will have the following form

$$\tilde{\mathbb{Y}} = \mathcal{O}_t x_{t-L+1} + \bar{H}^u_t \mathbb{U} + \bar{H}^d_t \mathbb{D} + \bar{H}^v_t \mathbb{V} + \mathbb{E}. \tag{56}$$

The linearization is done around a predicted trajectory, that is, given an estimate of the initial state in the window the trajectory is predicted in the window only using (55a).

If the fault is parameterized, it is assumed that the profile of the disturbance is a smooth function with respect to time. Thus, the disturbance could be parameterized as in Section 2.3. Each dimension in the disturbance vector is modeled separately, so $\bar{H}_t^{d^i}$ denotes the stacked system matrix for dimension i. This matrix is built up as (4) but using the i:th column of $\bar{H}_t^d$. The stacked system can now be written as

$$\tilde{\mathbb{Y}} = \mathcal{O}_t x_{t-L+1} + \sum_{i=1}^{n_d} \bar{H}_t^{d^i} \mathbb{D}^i + \bar{H}_t^v \mathbb{V} + \mathbb{E}, \tag{57}$$

where $\mathbb{D}^i = \left({d_{t-L+1}^i}^T, \dots, {d_t^i}^T \right)^T$. Each component of the disturbance is then modeled as

$$d_t^i = \phi_{i,t}^T \theta_i. \tag{58}$$

The influence of the disturbances can now be described as

$$\sum_{i=1}^{n_d} \bar{H}_t^{d^i} \mathbb{D}^i = \underbrace{\left(\bar{H}_t^{d^1} \cdots \bar{H}_t^{d^{n_d}} \right) \operatorname{diag}\left(\phi_{1,t}^T \cdots \phi_{n_d,t}^T \right)}_{\triangleq \bar{H}^\theta} \underbrace{\begin{pmatrix} \theta_1 \\ \vdots \\ \theta_{n_d} \end{pmatrix}}_{\triangleq \Theta}. \tag{59}$$

In the tests in Section 7.3, a third order basis is chosen for the fault. A plot of the basis functions are shown in Figure 1.

7.2 Detection Algorithm

The state estimation filter, including the disturbance detection, can be described as in Algorithm 1. The fault parameterization, described above can either be used or not in step 3.

Algorithm 1 Detection Filter

---

1. Time update according to an EKF.
2. Measurement update using non-disturbed sensors. This is done according to an EKF, but the measurement equation is limited to the set of non-disturbed sensors, $\mathcal{S}$. That is
$$y = \begin{pmatrix} \vdots \\ h_i(q_t) \\ \vdots \end{pmatrix}, \quad i \in \mathcal{S}. \tag{60}$$
3. Detection of disturbed sensors. Detection is done either according to the parity-space method or using the smoothing approach. The set $\mathcal{S}$ is updated accordingly.

7.3 Test Results

To show the performance of the disturbance detection algorithms described in previous chapter and to show the characteristics of the sensors, some measurement data is collected.

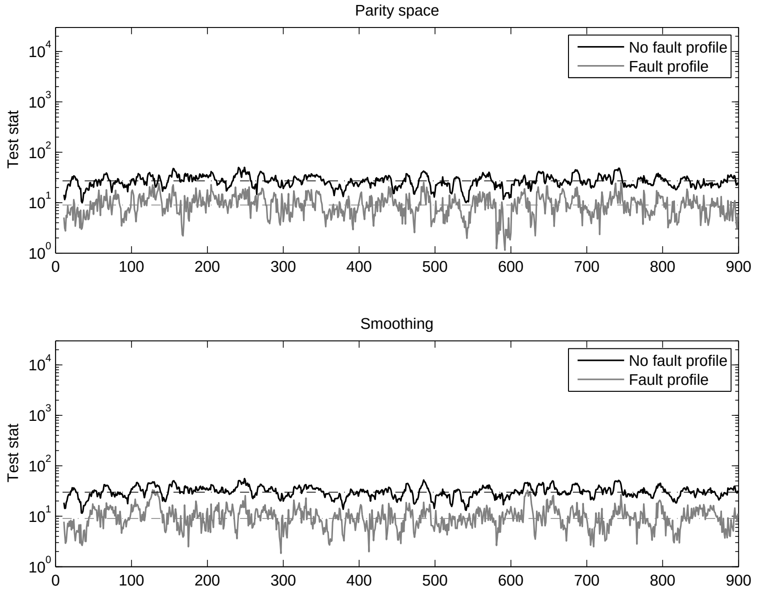
The data is collected under different circumstances, i.e., when the sensor is at rest, is slowly rotated, with and without disturbances. All data is collected with a sampling rate of 100 Hz and the length of the testing window is 10 samples in all tests. Table 1 presents the datasets.

Table 1: *Data sets collected using the* IMU.

Data Set	Disturbance	Movement
UD1	Undisturbed	At rest
D1	Magnetometer dist	At rest
UD2	Undisturbed	Rotated around e_x^b
D2	Magnetometer dist	Rotated around e_x^b

Sensor at Rest

The purpose of the first data set, UD1, is to explore the noise characteristics of the sensors. The IMU is here lying still at a table and data is collected during 9 seconds. The orientation of the sensor is set so that the e_z^b-axis is pointing outwards from the earth along the gravity field (perpendicular to the table). The test statistics are shown in Figure 5. The average of the test statistic should equal the degrees of freedom in the residual, see Section 5.2. This number is plotted as a dotted line in the figure and corresponds well to the actual outcome. Note that the fault profile lowers the average of the test statistic.


Parity space

Smoothing

Figure 5: *Test statistics for the data set UD1. Theoretical averages for undisturbed operation is plotted with dotted line.*

The experiment setup for data set D1 is similar to UD1. The difference is that the magnetometer is disturbed. The disturbance was caused by a ferro-magnetic object (a pair of scissors), which was moved towards the sensor. The closest distance was approximately 30 cm. The components of the magnetometer reading are plotted in Figure 6. The test statistics are shown in Figure 7, where the disturbance is clearly detected especially when smoothing is used. Note that the average for the test statistic with fault profile is lower during the undisturbed periods, but are equal during the disturbance since the fault profile can describe this smooth type of disturbance.

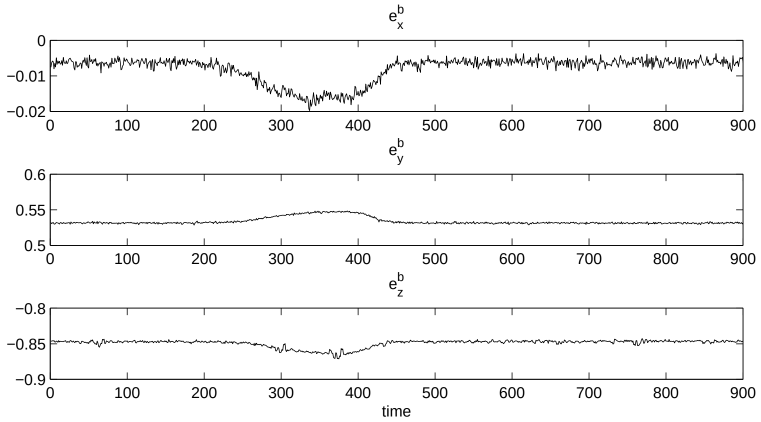


Figure 6: *The components of the disturbed magnetometer readings in data set D1.*

Rotating Sensor

To show the performance of detection when the sensor is rotating, data sets UD2 and D2 are collected when the IMU is rotated around the e^b_x-axis. The 3-dimensional magnetometer readings form a circle when it is not disturbed and a disturbance will make it deviate from the circle. The magnetometer readings are shown in Figure 8, where a disturbance can be seen to the right hand side in 8(b).

To accurately estimate the orientation of the sensor, the accelerometer measurements has to be disregarded during acceleration of the sensor as described in Section 7.2. Even though the sensor is just rotated in this example, the accelerometer show some acceleration since they are not centered in the sensor unit. The accelerometer is therefore ignored during most of the circular movement. Figure 9 shows the test statistics for data set UD2. It can be seen here that the actual averages are a little bit higher than the theoretical average. Possible explanations are linearization errors and misalignments in the sensor configuration. The disturbance in data set D2 is clearly seen in Figure 10 both with parity space and smoothing.

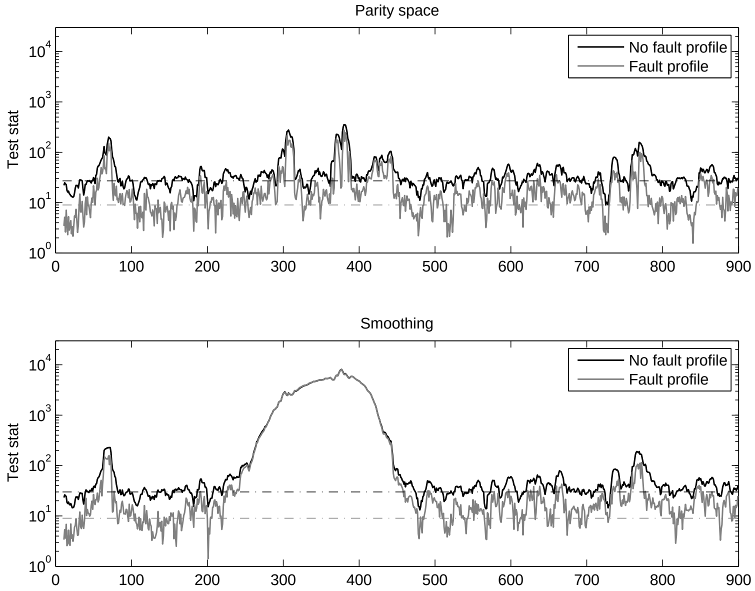
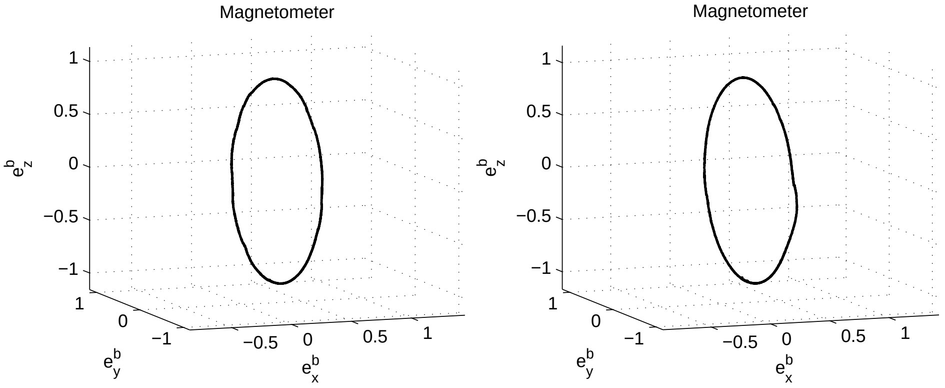


Figure 7: *Test statistics for data set D1, the disturbance is clearly visible when smoothing is used. Theoretical averages for undisturbed operation is plotted with dotted line.*



(a) *Undisturbed* ***(b)*** *Disturbed*

Figure 8: *Magnetometer readings from data set UD2 (a) and D2 (b). The light disturbance for D2 is visible as a small deviation from the circle in (b).*

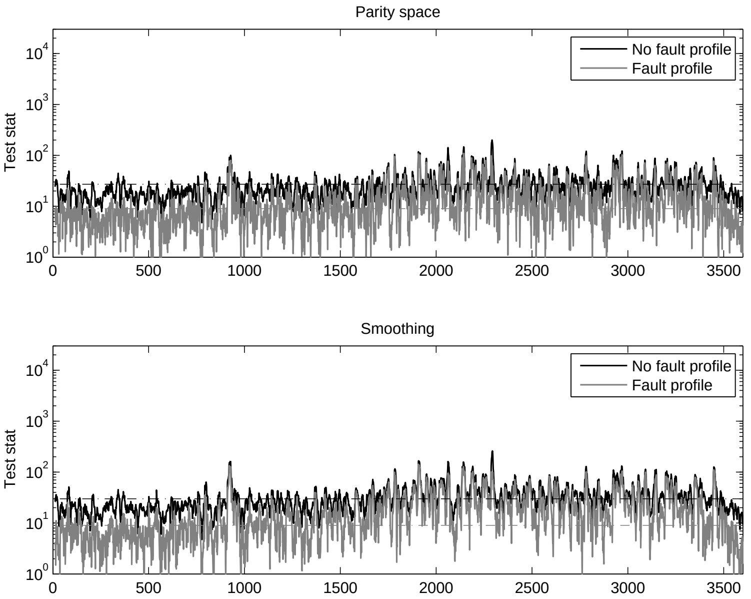


Figure 9: *Test statistic for data set UD2. Theoretical averages for undisturbed operation is plotted with dotted line.*

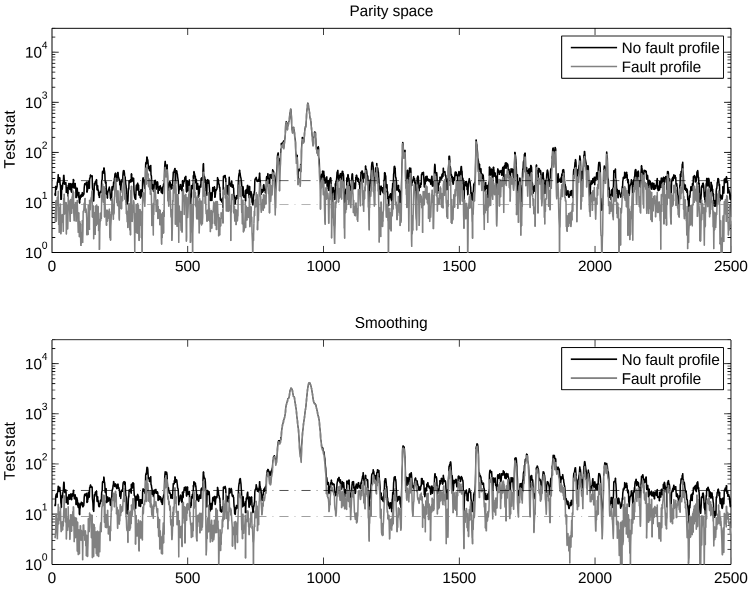


Figure 10: *Test statistic for data set D2. Theoretical averages for undisturbed operation is plotted with dotted line.*

8 Conclusions and Discussion

This work started with deriving parity-space residuals using different estimates of the initial state in a batch of data. This residual and its stochastic properties were then used for creating a GLR test. This test is highly related to the GLR test in Willsky and Jones (1976). This test is defined in terms of the state space model (1) for a step fault at an unknown time $t-L$. That is, the change time is unknown, which leads to a bank of filters, each one matched to a certain change time $t-L$. To limit computational complexity, the authors propose to only consider one fix change time $t-L$. The proposed algorithm consists of a nominal Kalman filter that runs to time t, in contrast to the Kalman filter with time lag in (23), and a matched filter that computes both an estimate of the fault and a state compensation term.

The advantage of parity space with smoothing is the geometrical interpretations and in particular its natural application to nonlinear systems. We have further pointed out how a smooth parameterization of the fault profile can improve the detection performance. Both simulations on a DC-motor and a real world example on fault detection on a magnetometer has shown to work well.

A Appendix

A.1 Proof of Lemma 1

If (35) is a positive definite matrix, the following must hold

$$x^T \begin{pmatrix} W_2^T & \mathcal{O} \end{pmatrix} \begin{pmatrix} S & 0 \\ 0 & Q \end{pmatrix} \begin{pmatrix} W_2 \\ \mathcal{O}^T \end{pmatrix} x > 0 \quad \forall x \neq 0. \tag{61}$$

Since it is known that S and Q are positive definite, it suffices to show that

$$x^T \begin{pmatrix} W_2^T & \mathcal{O} \end{pmatrix} \neq 0 \quad \forall x \neq 0. \tag{62}$$

The matrix W_2^T can be written as $I-\mathcal{O}M$ where M is an arbitrary matrix. If $x \in \mathcal{N}(\mathcal{O}^T)$ and $x \neq 0$ then

$$x^T \begin{pmatrix} W_2^T & \mathcal{O} \end{pmatrix} = \begin{pmatrix} x^T & 0 \end{pmatrix} \neq 0. \tag{63}$$

If $x \notin \mathcal{N}(\mathcal{O}^T)$ and $x \neq 0$ then

$$x^T \begin{pmatrix} W_2^T & \mathcal{O} \end{pmatrix} = \begin{pmatrix} x^T - x^T\mathcal{O}M & x^T\mathcal{O} \end{pmatrix} \neq 0 \tag{64}$$

since at least the second element is nonzero. Thus, we have shown that there is no $x \neq 0$ such that (62) is invalidated.

References

M. Abramowitz and I. A. Stegun, editors. *Handbook of Mathematical Functions: with Formulas, Graphs, and Mathematical Tables.* Dover Publications, 1965.

M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes: Theory and Application.* Information and system sciences series. Prentice Hall, Englewood Cliffs, NJ, 1993.

Å. Björck. *Numerical Methods for Least Squares Problems.* SIAM Publications, 1996.

A. Y. Chow and A. S. Willsky. Analytical redundancy and the design of robust failure detection systems. *IEEE Transactions on Automatic Control*, 29(7):603–614, 1984.

X. Ding, L. Guo, and T. Jeinsch. A characterization of parity space and its application to robust fault detection. *IEEE Transactions on Automatic Control*, 44(2):337–343, 1999.

M. Fouladirad and I. Nikiforov. Optimal statistical fault detection with nuisance parameters. *Automatica*, 41:1157–1171, 2005.

J. Gertler. Fault detection and isolation using parity relations. *Control Engineering Practice*, 5(5):653–661, 1997.

J. J. Gertler. *Fault Detection and Diagnosis in Engineering Systems.* Marcel Dekker, Inc., 1998.

G. H. Golub and C. F. van Loan. *Matrix Computations.* John Hopkins University Press, 3 edition, 1996. ISBN 0-2018-54-14-8.

F. Gustafsson. Statistical signal processing approaches to fault detection. *Annual Reviews in Control*, 31(1):41–54, 2007.

F. Gustafsson. *Adaptive filtering and change detection.* John Wiley & Sons, Ltd, 2 edition, 2001.

F. Gustafsson. Stochastic fault diagnosability in parity spaces. In *Proceedings of the 15th IFAC World Congress*, Barcelona, Spain, July 2002.

G. Hendeby and F. Gustafsson. Detection limits for linear non-Gaussian state-space models. In *6th IFAC Symposium on Fault Detection, Supervision and Safety of Technical Processes*, Beijing, P. R. China, Aug.–Sept. 2006.

S. M. Kay. *Fundamentals of Statistical Signal Processing: Detection Theory*, volume 2. Prentice-Hall, Inc, 1998. ISBN 0-13-504135-X.

J. B. Kuipers. *Quaternions and Rotation Sequences.* Princeton Univerity Press, 1999.

C. Meyer. *Matrix Analysis and Applied Linear Algebra.* SIAM, 2000.

T. J. Rivlin. *The Chebyshev Polynomials.* John Wiley & Sons, Inc, 1974.

D. Törnqvist. *Statistical Fault Detection with Applications to IMU Disturbances.* Licentiate thesis no. 1258, Department of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden, June 2006.

A. Willsky and H. Jones. A generalized likelihood ratio approach to the detection and estimation of jumps in linear systems. *IEEE Transactions on Automatic Control*, 21: 108–112, 1976.

Xsens Technologies B.V. Online: http://www.xsens.com, 2008.

Paper C

Detecting Spurious Features using Parity Space

Authors: David Törnqvist, Thomas B. Schön, Fredrik Gustafsson

Edited version of the paper: D. Törnqvist, T. B. Schön, and F. Gustafsson. Detecting spurious features using parity space. In *Proceedings of 10th International Conference on Control, Automation, Robotics and Vision*, Hanoi, Vietnam, Dec. 2008a. Accepted for publication

Preliminary version: Published as Technical Report LiTH-ISY-R-2864, Dept. of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

Detecting Spurious Features using Parity Space

David Törnqvist, Thomas B. Schön and Fredrik Gustafsson

Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
`{tornqvist, schon, fredrik}@isy.liu.se`

Abstract

Detection of spurious features is instrumental in many computer vision applications. The standard approach is feature based, where extracted features are matched between the image frames. This approach requires only vision, but is computer intensive and not yet suitable for real-time applications. We propose an alternative based on algorithms from the statistical fault detection literature. It is based on image data and an inertial measurement unit (IMU). The principle of analytical redundancy is applied to batches of measurements from a sliding time window. The resulting algorithm is fast and scalable, and requires only feature positions as inputs from the computer vision system. It is also pointed out that the algorithm can be extended to also detect non-stationary features (moving targets for instance). The algorithm is applied to real data from an unmanned aerial vehicle in a navigation application.

1 Introduction

Computer vision algorithms are today used in a large number of applications within areas of tracking, navigation, augmented reality and simultaneous localization and mapping (SLAM). In common for these applications is that features in the image frames are detected and used as observations in a (Kalman) filter to estimate camera pose or target bearings. For a feature to be useful, it has to be stationary in space and clearly shaped so its position is unambiguously computed in consecutive frames by the feature detection algorithm. If the latter is not the case, the feature is called a spurious feature.

The classical approach is a stand-alone computer vision feature-based solution. One standard approach for avoiding spurious features is to use the so called random sample consensus (RANSAC) algorithm Fischler and Bolles (1981). A problem with this algorithm is that it is not directly suitable for real-time algorithms. Some interesting recent work on reducing the computational complexity have been presented in Nistér (2005).

To mention a few applications, where RANSAC type outlier detection is used we have Kanade et al. (2004), Maimone et al. (2007), Nistér et al. (2006). For image to image comparisons the difference between the predicted feature location and its actual location can be used to detect outliers. Another idea is to make use of the epipolar geometry to detect outliers, simply by computing the distance to the epipolar lies for each correspondance. Is this distance is larger than a certain threshold the feature is detected as a spurious feature Torr et al. (1995).

For real-time applications with limited computational resources, methods based on correlation maximization of patch templates in each frame for a large number of features should be avoided.

The approach we suggest is prediction based, combining IMU measurements with a model to obtain the predictions. The detection part is an application of the stochastic parity space approach in fault detection. The stochastic parity space approach as described in Gustafsson (2007) extends the classical parity space approach in fault detection Chow and Willsky (1984) to stochastic measurement errors and process noise. The proposed algorithm consists of the following components:

- An IMU for predicting how the features move from frame to frame.
- A list of feature points for each frame is used as measurement vector. Here, for instance, the Harris detector Harris and Stephens (1988) can be used.
- A sliding window where the feature points are monitored for outliers.
- A parity space approach where the feature observations are projected onto a smaller subspace which eliminates unknown inputs, attenuates the noise and highlights outliers.

In a way, the analytical redundancy in the parity space extends the concept of projective geometry to a sequence of frames, utilizing a temporal motion model for the camera pose.

The algorithm is applied to airborne images in a SLAM application, where the goal is to navigate an unmanned aerial vehicle (UAV) back to home-base without infrastructure, as described in Karlsson et al. (2008). This case is particularly challenging, since the ground scene is rich of spurious features. Yet, the algorithm is quite successful in detecting spurious features.

The outline is as follows. Section 2 surveys the stochastic parity space approach to fault detection in linear systems. Section 3 presents a kinematic model suitable for fusion of IMU and camera information, and gives a state space model suitable for the parity space approach. Section 4 illustrates the method on artificial data from the realistic scenario, on which the method is evaluated on real data in Section 5.

2 Parity Space

If a model of a system is available, it is possible to determine in which part of the output space an output from a healthy system can reside. If there is signal energy in other parts of the output space, it is not in accordance with the model and a fault must therefore be present. This is exploited within the parity space framework Chow and Willsky (1984), where an orthogonal projection of the output is used to create a residual that is non-sensitive to healthy outputs.

2.1 Modeling

Let us consider a discrete-time dynamical system in the form

$$x_{t+1} = F_t x_t + G_t^u u_t + G_t^f f_t + G_t^v v_t, \tag{1a}$$

$$y_t = H_t x_t + H_t^u u_t + H_t^f f_t + e_t, \tag{1b}$$

where x_t is the state vector, u_t an input signal, f_t a fault vector, y_t a measurement, v_t process noise and e_t the measurement noise. Since we are interested in monitoring how the system behaves over time, the system can be written in batch form. The batch form is often used in fault detection and diagnosis Chow and Willsky (1984), Gertler (1998), Gustafsson (2000). Stack L signal values to define the batched signal vectors like $\mathbb{Y} = \left(y_{t-L+1}^T, \dots, y_t^T\right)^T$, for all signals. Then, the outputs in that window will be given by

$$\mathbb{Y} = \mathcal{O}_t x_{t-L+1} + \bar{H}_t^u \mathbb{U} + \bar{H}_t^f \mathbb{F} + \bar{H}_t^v \mathbb{V} + \mathbb{E}, \tag{2}$$

with the extended observability matrix

$$\mathcal{O}_t = \begin{pmatrix} H_{t-L+1} \\ H_{t-L+2} F_{t-L+1} \\ \vdots \\ H_t \prod_{k=t-1}^{t-L+1} F_k \end{pmatrix}. \tag{3}$$

The matrices determining how the remaining signals affect the system are described by

$$\bar{H}_t^s = \begin{pmatrix} H_{t-L+1}^s & 0 & \cdots & 0 \\ H_{t-L+2} G_{t-L+1}^s & H_{t-L+2}^s & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ H_t \prod_{k=t-1}^{t-L+2} F_k G_{t-L+1}^s & \cdots & H_t G_{t-1}^s & H_t^s \end{pmatrix}, \tag{4}$$

where $s = \{u, f, v\}$. For simplicity, denote $\mathbb{Z} = \mathbb{Y} - \bar{H}_t^u \mathbb{U}$, implying that the system (2) can be written as

$$\mathbb{Z} = \mathcal{O}_t x_{t-L+1} + \bar{H}_t^f \mathbb{F} + \bar{H}_t^v \mathbb{V} + \mathbb{E}. \tag{5}$$

If the system can be written in the form (5), where $\mathsf{Cov}(\mathbb{V} + \mathbb{E}) = S$, we can compute an estimate of the initial state according to

$$\hat{x}_{t-L+1} = \mathcal{O}_t^\dagger \mathbb{Z}. \tag{6}$$

In this work, the standard parity space approach with orthogonal projection is used. This implies that the initial state in the test window is estimated using the pseudo-inverse as shown above. An interesting extension would be to make a minimum variance estimate of the initial state which corresponds to an oblique projection in the parity space framework, see Törnqvist and Gustafsson (2006). Note that in the special case $S = \sigma I$, (6) is in fact the minimum variance estimate.

2.2 Residual Generation

A residual for detecting abnormal behavior of the output y_t can now be formed as

$$\tilde{r} = \mathbb{Z} - \mathcal{O}_t \hat{x}_{t-L+1} = \mathbb{Z} - \underbrace{\mathcal{O}_t \mathcal{O}_t^{\dagger}}_{\mathcal{P}_{\mathcal{O}_t}} \mathbb{Z} = \underbrace{(I - \mathcal{P}_{\mathcal{O}_t})}_{\mathcal{P}_{\mathcal{O}_t^{\perp}}} \mathbb{Z}. \tag{7}$$

Since $\mathcal{P}_{\mathcal{O}_t^{\perp}}$ is rank deficient, the information in $\tilde{r}$ can be expressed using a lower dimensional residual

$$r = \mathcal{B}_{\mathcal{O}^{\perp}}^{T} \mathbb{Z}, \tag{8}$$

where the columns of $\mathcal{B}_{\mathcal{O}^{\perp}}$ forms an orthonormal basis for $\mathcal{R}(\mathcal{O}^{\perp})$ (This can be computed with an SVD). The residual r has the covariance

$$\mathsf{Cov}(r) = \mathcal{B}_{\mathcal{O}^{\perp}}^{T} \, S \, \mathcal{B}_{\mathcal{O}^{\perp}} \, . \tag{9}$$

A normalized residual is then given by

$$\bar{r} = \underbrace{(\mathcal{B}_{\mathcal{O}^{\perp}}^{T} \, S \, \mathcal{B}_{\mathcal{O}^{\perp}})^{-1/2} \, \mathcal{B}_{\mathcal{O}^{\perp}}^{T}}_{\triangleq W^T} \mathbb{Z}. \tag{10}$$

2.3 Statistical tests

To test whether a fault has occurred or not, the following hypothesis test is formed,

$$\mathcal{H}_0 : \ \bar{r} \sim N(0, I), \tag{11a}$$

$$\mathcal{H}_1 : \ \bar{r} \sim N(W^T \bar{H}^f \mathbb{F}, I). \tag{11b}$$

To decide between the hypotheses, a GLR test will be performed using the following test statistic

$$L = 2 \log \left(\frac{\sup_{\mathbb{F}} p(\bar{r} - W^T \bar{H}^f \mathbb{F})}{p(\bar{r})} \right) = \sup_{\mathbb{F}} 2 \log \frac{\frac{1}{(2\pi)^{n/2}} e^{-\frac{1}{2} \|\bar{r} - W^T \bar{H}^f \mathbb{F}\|_2^2}}{\frac{1}{(2\pi)^{n/2}} e^{-\frac{1}{2} \|\bar{r}\|_2^2}}$$
$$= \sup_{\mathbb{F}} -\|\bar{r} - W^T \bar{H}^f \mathbb{F}\|_2^2 + \|\bar{r}\|_2^2, \tag{12}$$

where n denotes the dimension of $\bar{r}$. This function is maximized for $\mathbb{F} = (W^T \bar{H}^f)^{\dagger} \bar{r}$, which gives

$$L = -\|\bar{r} - \underbrace{W^T \bar{H}^f (W^T \bar{H}^f)^{\dagger}}_{\mathcal{P}_{W^T \bar{H}^f}} \bar{r}\|_2^2 + \|\bar{r}\|_2^2 = \bar{r}^T \mathcal{P}_{W^T \bar{H}^f} \bar{r}. \tag{13}$$

To choose suitable thresholds for the test statistics above, it is necessary to compute their distributions. While having Gaussian noise, the test statistics will be chi-square distributed variables. For a thorough review of statistics in signal processing, see e.g., Kay (1998). The normalized residual is distributed as

$$\bar{r} \sim \mathsf{N}(W^T \bar{H}^f \mathbb{F}, I), \tag{14}$$

where $\mathbb{F} = 0$ under the null hypothesis (11). The test statistic is then distributed as the non-central χ^2-distribution

$$L = \bar{r}^T \mathcal{P}_{W^T \bar{H}^f} \bar{r} \sim \chi'^2_\nu(\lambda), \tag{15}$$

where $\nu = \operatorname{rank}(W^T \bar{H}^f)$ and

$$\lambda = (W^T \bar{H}^f \mathbb{F})^T \mathcal{P}_{W^T \bar{H}^f} W^T \bar{H}^f \mathbb{F} = (W^T \bar{H}^f \mathbb{F})^T W^T \bar{H}^f \mathbb{F}. \tag{16}$$

Note that $\lambda = 0$ in the fault-free case and then the test statistic is distributed according to the central χ^2-distribution $L \sim \chi^2_\nu$. The threshold is then chosen from the χ^2-distribution so that the fault-free hypothesis is rejected erroneously only with a small probability.

3 Detection of Spurious Features

In this work we assume that stationary features are observed through a pinhole camera. As states in the model, the three-dimensional position of the feature $m_{i,t}$ is used and the full state vector is $m_t = \begin{pmatrix} m^T_{1,t}, \dots, m^T_{n_m,t} \end{pmatrix}^T$. The pinhole camera Ma et al. (2006), Hartley and Zisserman (2003) is assumed to be calibrated and have unit focal length. The system equations can be written as

$$m_{t+1} = m_t, \tag{17a}$$

$$y_{i,t} = \underbrace{\frac{1}{\mathsf{z}^c_{i,t}} \begin{pmatrix} \mathsf{x}^c_{i,t} \\ \mathsf{y}^c_{i,t} \end{pmatrix}}_{h_i(m^c_{i,t}) = h_i(m_{i,t}, p_t)} + e_{i,t} = \frac{1}{\mathsf{z}^c_{i,t}} \begin{pmatrix} I & 0 \end{pmatrix} R_t(m_{i,t} - p_t) + e_{i,t}, \tag{17b}$$

where

$$m^c_{i,t} = \begin{pmatrix} \mathsf{x}^c_{i,t} \\ \mathsf{y}^c_{i,t} \\ \mathsf{z}^c_{i,t} \end{pmatrix} = R_t(m_{i,t} - p_t). \tag{18}$$

Here, R_t is the rotation matrix between the world and the camera coordinate frame, p_t is the translation between the coordinate frames and $m^c_{i,t}$ the feature position in the camera coordinate frame. This model could be extended to include dynamics for m_t if we want to model moving objects as well. To use the parity space framework, the measurement equation is simply linearized according to,

$$y_{i,t} = h_i(m^c_{i,t}) + e_{i,t} \approx h_i(\hat{m}^c_t) + H_{i,t}(m^c_{i,t} - \hat{m}^c_{i,t}) + e^c_{i,t}, \tag{19}$$

for $j = 1, \dots, n_m$, where $e^c_{i,t}$ is the noise from the camera sensor and the Jacobian matrix $H_{i,t}$ is

$$H_{i,t} = \left. \frac{\partial h_i(m^c_{i,t})}{\partial m^c_{i,t}} \right|_{m^c_{i,t} = \hat{m}^c_{i,t}} = \frac{1}{\hat{\mathsf{z}}^c_{i,t}} \begin{pmatrix} 1 & 0 & -\frac{\hat{\mathsf{x}}^c_{i,t}}{\hat{\mathsf{z}}^c_{i,t}} \\ 0 & 1 & -\frac{\hat{\mathsf{y}}^c_{i,t}}{\hat{\mathsf{z}}^c_{i,t}} \end{pmatrix}. \tag{20}$$

Note that $\hat{m}^c_{i,t} = \begin{pmatrix}\hat{\mathsf{x}}^c_{i,t} & \hat{\mathsf{y}}^c_{i,t} & \hat{\mathsf{z}}^c_{i,t}\end{pmatrix}^T$ denotes the one-step ahead prediction from the filter. Evaluation of (19) gives

$$\begin{aligned} y_{i,t} &\approx h_i(\hat{m}^c_{i,t}) + H_{i,t}(m^c_{i,t} - \hat{m}^c_{i,t}) + e_{i,t} \\ &= \frac{1}{\hat{\mathsf{z}}^c_{i,t}} \begin{pmatrix}\mathsf{x}^c_{i,t} \\ \mathsf{y}^c_{i,t}\end{pmatrix} + \underbrace{\frac{\hat{\mathsf{z}}^c_{i,t} - \mathsf{z}^c_{i,t}}{\hat{\mathsf{z}}^c_{i,t}} \begin{pmatrix}\hat{\mathsf{x}}^c_{i,t}/\hat{\mathsf{z}}^c_{i,t} \\ \hat{\mathsf{y}}^c_{i,t}/\hat{\mathsf{z}}^c_{i,t}\end{pmatrix}}_{e^{\mathsf{z}}_{i,t}} + e^c_{i,t}. \end{aligned} \tag{21}$$

The term $e^{\mathsf{z}}_{i,t}$ has zero mean and covariance

$$\mathsf{Cov}(e^{\mathsf{z}}_{i,t}) = \frac{P_{\mathsf{z}^c_{i,t}}}{(\hat{\mathsf{z}}^c_{i,t})^2} \begin{pmatrix}\hat{\mathsf{x}}^c_{i,t}/\hat{\mathsf{z}}^c_{i,t} \\ \hat{\mathsf{y}}^c_{i,t}/\hat{\mathsf{z}}^c_{i,t}\end{pmatrix} \begin{pmatrix}\hat{\mathsf{x}}^c_{i,t}/\hat{\mathsf{z}}^c_{i,t} \\ \hat{\mathsf{y}}^c_{i,t}/\hat{\mathsf{z}}^c_{i,t}\end{pmatrix}^T, \tag{22}$$

where $P_{\mathsf{z}^c_{i,t}} = \mathsf{Cov}(\mathsf{z}^c_{i,t})$. Hence, we can make the approximation that this term is only a contribution to the measurement noise. Define the total measurement noise as $e_{i,t} = e^{\mathsf{z}}_{i,t} + e^c_{i,t}$. Using this approximation, the measurement equation can be written as

$$y_{i,t} \approx \frac{1}{\hat{\mathsf{z}}^c_{i,t}} \begin{pmatrix}\mathsf{x}^c_{i,t} \\ \mathsf{y}^c_{i,t}\end{pmatrix} + e_{i,t} = \frac{1}{\hat{\mathsf{z}}^c_{i,t}} \begin{pmatrix}I & 0\end{pmatrix} R_t(m_{i,t} - p_t) + e_{i,t}. \tag{23}$$

Now, define $\hat{Z} = \mathrm{diag}(\hat{\mathsf{z}}^c_{1,t} I_2, \hat{\mathsf{z}}^c_{2,t} I_2, \ldots)$ and premultiply the total measurement equation with this matrix. This equation can then be written as

$$\hat{Z} y_t = \underbrace{\mathrm{diag}\left(\begin{pmatrix}I & 0\end{pmatrix} R_t, \begin{pmatrix}I & 0\end{pmatrix} R_t, \ldots\right)}_{H_t} m_t - \underbrace{\begin{pmatrix}\begin{pmatrix}I & 0\end{pmatrix} R_t \\ \begin{pmatrix}I & 0\end{pmatrix} R_t \\ \vdots\end{pmatrix}}_{H^u_t} \underbrace{p_t}_{u_t} + \hat{Z} e_t, \tag{24}$$

This linearized model can be used in the parity space framework from Section 2, use the notation $x_t = m_t$, $F_t = I$ and the rest of the definitions in (24).

3.1 Known Translation

In case the translation is known, it is modeled as a deterministic input. The system can be written in batch form (2), but using $\mathbb{Y}$ as stacked $\hat{Z} y_t$ and observing that $\bar{H}^v \mathbb{V} = 0$ since the system has no process noise (17),

$$\mathbb{Y} = \mathcal{O} x_{t-L+1} + \bar{H}^u \mathbb{U} + \bar{H}^f \mathbb{F} + \mathbb{E}. \tag{25}$$

The residual is then formed according to Section 2.2

$$\bar{r}_1 = \underbrace{(\mathcal{B}^T_{\mathcal{O}^\perp} S \mathcal{B}_{\mathcal{O}^\perp})^{-1/2} \mathcal{B}^T_{\mathcal{O}^\perp}}_{\triangleq W^T} \mathbb{Z}, \tag{26}$$

where $\mathbb{Z} = \mathbb{Y} - \bar{H}^u\mathbb{U}$. For $n_m = 3$, $L = 3$ and $R_0 = I$ (without loss of generality), $\mathcal{O}$ will have the following form

$$\mathcal{O} = \begin{pmatrix} H_{t-2} \\ H_{t-1} \\ H_t \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} I & 0 \end{pmatrix} & 0 & 0 \\ 0 & \begin{pmatrix} I & 0 \end{pmatrix} & 0 \\ 0 & 0 & \begin{pmatrix} I & 0 \end{pmatrix} \\ \begin{pmatrix} I & 0 \end{pmatrix} R_1 & 0 & 0 \\ 0 & \begin{pmatrix} I & 0 \end{pmatrix} R_1 & 0 \\ 0 & & \begin{pmatrix} I & 0 \end{pmatrix} R_1 \\ \begin{pmatrix} I & 0 \end{pmatrix} R_2 & 0 & 0 \\ 0 & \begin{pmatrix} I & 0 \end{pmatrix} R_2 & 0 \\ 0 & 0 & \begin{pmatrix} I & 0 \end{pmatrix} R_2 \end{pmatrix}. \tag{27}$$

The rank of this matrix is $3 \cdot 3 = 9$ and it is easy to see that the rank in the general case is $\text{rank}(\mathcal{O}) = 3n_m$. The number of rows of $\mathcal{O}$ is $2n_mL$, so the dimension of the orthogonal column-space $\mathcal{O}^\perp$ is $2n_mL - 3n_m = (2L - 3)n_m$. This is also the dimension of the residual $\bar{r}_1$ in (26).

3.2 Unknown Translation

When the translation is unknown, the predictions of the feature position in the camera coordinate frame will become more uncertain. The covariance in (22) depends on these predictions so in order to limit this we will make the following pragmatic approximation

$$\mathsf{Cov}(e^z_{i,t}) = \frac{P_{\mathsf{z}^c_{i,t}}}{(\hat{\mathsf{z}}^c_{i,t})^2} \begin{pmatrix} \hat{\mathsf{x}}^c_{i,t}/\hat{\mathsf{z}}^c_{i,t} \\ \hat{\mathsf{y}}^c_{i,t}/\hat{\mathsf{z}}^c_{i,t} \end{pmatrix} \begin{pmatrix} \hat{\mathsf{x}}^c_{i,t}/\hat{\mathsf{z}}^c_{i,t} \\ \hat{\mathsf{y}}^c_{i,t}/\hat{\mathsf{z}}^c_{i,t} \end{pmatrix}^T \approx \frac{P_z}{(\hat{\mathsf{z}}^c_{i,t})^2} y_{i,t} y^T_{i,t}. \tag{28}$$

This can be motivated by the fact that the sensor noise is fairly small and this noise term depends on the position of the feature in the image, see section 4 for an illustration. In the detection problem, the translation can no longer be seen as a known input to the system and the residual should therefore be insensitive to it. Hence, the batched system is rewritten as

$$\mathbb{Y} = \mathcal{O}x + \bar{H}^u\mathbb{U} + \bar{H}^f\mathbb{F} + \mathbb{E} = \underbrace{\begin{pmatrix} \mathcal{O} & \bar{H}^u \end{pmatrix}}_{\tilde{\mathcal{O}}} \begin{pmatrix} x \\ \mathbb{U} \end{pmatrix} + \bar{H}^f\mathbb{F} + \mathbb{E}. \tag{29}$$

A residual can now be constructed by an orthogonal projection onto $\tilde{\mathcal{O}}$ as

$$\bar{r}_2 = \underbrace{(\mathcal{B}^T_{\tilde{\mathcal{O}}^\perp}\, S\, \mathcal{B}_{\tilde{\mathcal{O}}^\perp})^{-1/2}\, \mathcal{B}^T_{\tilde{\mathcal{O}}^\perp}}_{\triangleq \tilde{W}^T}\, \mathbb{Y}, \tag{30}$$

where the columns of the orthonormal matrix $\mathcal{B}_{\tilde{\mathcal{O}}^\perp}$ spans the orthogonal image space to $\tilde{\mathcal{O}}$. For $n_m = 3$, $L = 3$ and $R_0 = I$ (without loss of generality), $\tilde{\mathcal{O}}$ would have the following form

$$\tilde{\mathcal{O}} = \begin{pmatrix} \mathcal{O} & \mathcal{O}_2 \end{pmatrix} \tag{31}$$

where $\mathcal{O}$ was given in (27) and

$$\mathcal{O}_2 = \begin{pmatrix} \begin{pmatrix} I & 0 \end{pmatrix} & 0 & 0 \\ \begin{pmatrix} I & 0 \end{pmatrix} & 0 & 0 \\ \begin{pmatrix} I & 0 \end{pmatrix} & 0 & 0 \\ 0 & \begin{pmatrix} I & 0 \end{pmatrix} R_1 & 0 \\ 0 & \begin{pmatrix} I & 0 \end{pmatrix} R_1 & 0 \\ 0 & \begin{pmatrix} I & 0 \end{pmatrix} R_1 & 0 \\ 0 & 0 & \begin{pmatrix} I & 0 \end{pmatrix} R_2 \\ 0 & 0 & \begin{pmatrix} I & 0 \end{pmatrix} R_2 \\ 0 & 0 & \begin{pmatrix} I & 0 \end{pmatrix} R_2 \end{pmatrix} \tag{32}$$

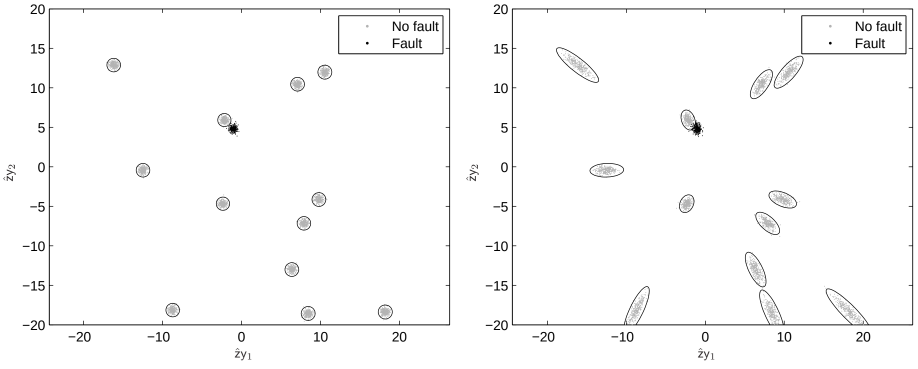
Using row operations, (32) can be transformed into

$$\begin{pmatrix} 0 & 0 & 0 & \begin{pmatrix} I & 0 \end{pmatrix} & 0 & 0 \\ \begin{pmatrix} I & 0 \end{pmatrix} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \begin{pmatrix} I & 0 \end{pmatrix} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \begin{pmatrix} I & 0 \end{pmatrix} R_1 & 0 \\ \begin{pmatrix} I & 0 \end{pmatrix} R_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & & \begin{pmatrix} I & 0 \end{pmatrix} R_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \begin{pmatrix} I & 0 \end{pmatrix} R_2 \\ \begin{pmatrix} I & 0 \end{pmatrix} R_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \begin{pmatrix} I & 0 \end{pmatrix} R_2 & 0 & 0 & 0 \end{pmatrix} \tag{33}$$

The rank of this matrix is $3 \cdot 2 + 2 \cdot 3 = 12$, or in the general case $3(n_m - 1) + 2L$. Since the number of rows are $2n_m L$, the dimension of the orthogonal image space to $\tilde{\mathcal{O}}$ will be $2n_m L - 3(n_m - 1) + 2L = (2L - 3)n_m - 2L + 3$. This will also be the dimension of the residual $\bar{r}_2$ in (30).

4 Simulations

For the simulations, a map of features similar to the real flight scenario in Section 5 is created. The scenario is a helicopter flying at an altitude of 58 m, with a downward-looking camera attached to it. The features are uniformly spread out over an area of 40×40 m and the altitude is generated from a Gaussian distribution $\mathsf{N}(58, 1)$. A fault is introduced with the direction $k\begin{pmatrix} -1 & 1 \end{pmatrix}^T$ for one of the features, the magnitude k is varied between the values $\{0, 0.02, 0.03\}$. The measurement noise is distributed as $e_{i,t} \sim \mathsf{N}(0, 0.005^2)$. Figure 1 shows 300 measurements $\hat{Z}y_t$ as in (24), both with exact knowledge of $\mathsf{z}^c_{i,t}$ ($\mathsf{Cov}(\mathsf{z}^c_{i,t}) = 0$) and when there is uncertainty present ($\mathsf{Cov}(\mathsf{z}^c_{i,t}) = 3^2$). The difference between these cases is if the noise term $e^{\mathsf{z}}_{i,t}$ in (22) is present or not. A faulty measurement is also shown, where the fault magnitude is 0.02. It can be noted that the measurement noise is increased radially when $\mathsf{Cov}(\mathsf{z}^c_{i,t})$ is increased. To make diagnosis of spurious features possible, the features are divided into groups of three. Detection is then made for each group and if a fault is detected, the measurements from the group are discarded.

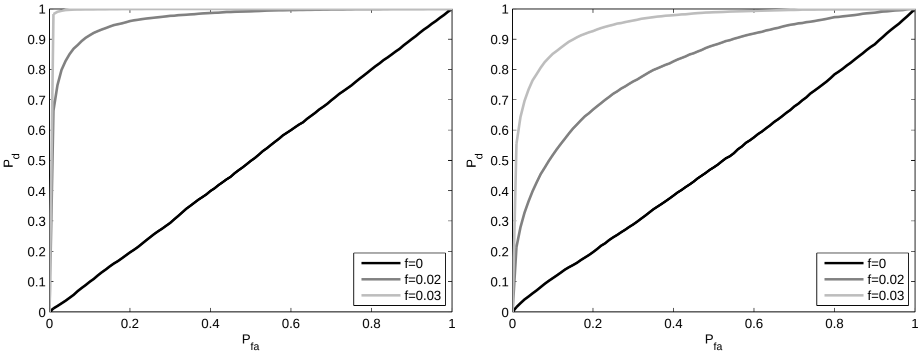


(a) $\mathsf{Cov}(\mathsf{z}^c_{i,t}) = 0$, *i.e.*, $\mathsf{z}^c_{i,t} = \hat{\mathsf{z}}^c_{i,t}$ ***(b)*** $\mathsf{Cov}(\mathsf{z}^c_{i,t}) = 3^2$

Figure 1: *Simulated measurements* $\hat{Z}y_t$ *of the features are shown (300 realizations). The ellipsoids show the confidence region corresponding to 3 standard deviations.*

4.1 Known Translation

If the translation is known, the dimension of the residual will be $(2L - 3)n_m = 3$. This will also be the degrees of freedom in the χ^2-test used for detection, see Section 2.3. In Figure 2, the Receiver Operation Characteristics (ROC) curves are plotted. The ROC-curve shows how the probability of detection P_d for a fault varies versus the probability of false alarm P_fa. The ROC-curve is a straight line for the no fault case. This is natural, since $P_\text{d} = P_\text{fa}$ for any threshold if there is nothing to detect. It can be noted that the probability of detection decreases with uncertainty of z^c. This is due to the increased measurement noise, illustrated in Figure 1.

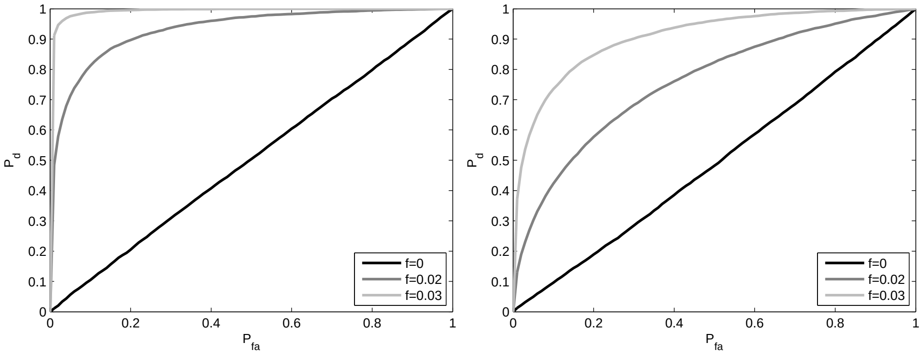


(a) $\mathsf{Cov}(\mathsf{z}^c_{i,t}) = 0$, *i.e.*, $\mathsf{z}^c_{i,t} = \hat{\mathsf{z}}^c_{i,t}$ ***(b)*** $\mathsf{Cov}(\mathsf{z}^c_{i,t}) = 3^2$

Figure 2: *ROC-curves showing the detection performance with known translation.*

4.2 Unknown Translation

In the case of unknown translation, the degrees of freedom in the χ^2-test is decreased to $(2L-3)n_m - 2L + 3 = 2$. This somewhat reduces the power of the test which can be seen in Figure 3. Given a certain P_{fa}, the P_{d} is lower here compared to the case of known translation in Figure 2. The strength with this test is that no knowledge of the translation is needed. In reality, the estimate of the position change between two frames is often poor.



(a) $\mathsf{Cov}(\mathsf{z}^c_{i,t}) = 0$, *i.e.*, $\mathsf{z}^c_{i,t} = \hat{\mathsf{z}}^c_{i,t}$

(b) $\mathsf{Cov}(\mathsf{z}^c_{i,t}) = 3^2$

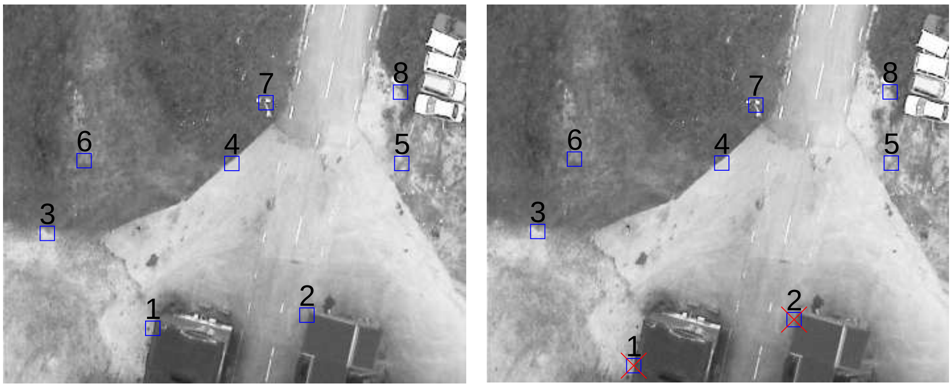
Figure 3: *ROC-curves showing the detection performance with unknown translation.*

5 Real World Experiment

In this example, simultaneous localization and mapping (SLAM) is performed using a Yamaha RMAX helicopter with an IMU and a camera looking at the ground. The task is to be able to localize the helicopter by combining inertial measurements with the camera image. During the process, landmarks are found in the image and the 3D positions of these are estimated. The setup and the SLAM algorithm are explained in Karlsson et al. (2008).

The detection of spurious features within the SLAM framework is the focus of the present work. Features in two consecutive frames are compared. The IMU computes the change in rotation of the camera between the frames, but the translation is considered unknown since the integrated accelerations gives high uncertainty. The theory derived in Section 3.2 can therefore be applied. To form the test, the features are divided into several groups. If a test indicates that there is an inconsistency in one of the groups, these features will be discarded.

The image sequence and IMU data that we have used for this paper are collected when flying a 300 m distance. Without using this detection method, the estimated position diverges, since spurious features contaminate the filter. When spurious features are detected, the position is estimated without problems. Figure 4 shows an example of when spurious features are detected. The eight features in view are grouped into three groups and a consistency test between image one and two is performed for each group. In this example, the group consisting of feature 1 and 2 is not consistent and the features are therefore discarded. Feature 1 is detected on an edge and the feature tracker makes a common mistake, which is to move the feature along this edge.



(a) *Features detected in the first image.*

(b) *Features detected in the second image, the group consisting of feature 1 and 2 are detected as inconsistent with the first image.*

Figure 4: *Example of spurious feature detection.*

6 Conclusions and Future Work

The contribution of this paper has been to show how the parity space framework from the statistical fault detection literature can be applied to find spurious features. The characteristics and performance have been shown in simulation examples. Furthermore, it has also been successfully implemented in a real-world example where it is necessary to estimate the helicopter position.

Acknowledgment

This work was supported by the strategic research center MOVIII, funded by the Swedish Foundation for Strategic Research, SSF.

References

A. Y. Chow and A. S. Willsky. Analytical redundancy and the design of robust failure detection systems. *IEEE Transactions on Automatic Control*, 29(7):603–614, 1984.

M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24 (6):381–395, 1981.

J. J. Gertler. *Fault Detection and Diagnosis in Engineering Systems.* Marcel Dekker, Inc., 1998.

F. Gustafsson. *Adaptive Filtering and Change Detection.* John Wiley & Sons, Ltd, New York, USA, 2000.

F. Gustafsson. Statistical signal processing approaches to fault detection. *Annual Reviews in Control*, 31(1):41–54, 2007.

C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference*, pages 147–151, Manchester, UK, 1988.

R. Hartley and A. Zisserman. *Multiple View Geometry in computer vision.* Cambridge University Press, Cambridge, UK, 2 edition, 2003.

T. Kanade, A. Amidi, and Q. Ke. Real-time and 3D vision for autonomous small and micro air vehicles. In *Proceedings of the 43rd Conference on Decision and Control (CDC)*, Paradise Island, Bahamas, Dec. 2004.

R. Karlsson, T. B. Schön, D. Törnqvist, G. Conte, and F. Gustafsson. Utilizing model structure for efficient simultaneous localization and mapping for a UAV application. In *Proceedings of IEEE Aerospace Conference*, 2008.

S. M. Kay. *Fundamentals of Statistical Signal Processing: Detection Theory*, volume 2. Prentice-Hall, Inc, 1998. ISBN 0-13-504135-X.

Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry. *An invitation to 3-D vision – from images to geometric models.* Interdisciplinary Applied Mathematics. Springer-Verlag, 2006.

M. Maimone, Y. Cheng, and L. Matthies. Two years of visual odometry on the mars exploration rovers. *Journal of Field Robotics*, 24(3):169–186, 2007.

D. Nistér. Preemptive RANSAC for live structure and mostion estimation. *Machine Vision and Applications*, 16(5):321–329, Dec. 2005.

D. Nistér, O. Neroditsky, and J. Bergen. Visual odometry for ground vehicle applications. *Journal of Field Robotics*, 23(1):3–20, 2006.

D. Törnqvist and F. Gustafsson. Eliminating the initial state for the generalized likelihood ratio test. In *Proceedings of IFAC Symposium SAFEPROCESS*, Beijing, China, Aug. 2006.

P. H. S. Torr, A. Zisserman, and S. J. Maybank. Robust detection of degenerate configurations for the fundamental matrix. In *Proceedings of the 5th International Conferences on Computer Vision*, pages 1037–1042, Boston, USA, 1995.

Paper D

Fast Particle Filters for Multi-Rate Sensors

Authors: Thomas B. Schön, David Törnqvist and Fredrik Gustafsson

Edited version of the paper: T. Schön, D. Törnqvist, and F. Gustafsson. Fast particle filters for multi-rate sensors. In *The 15th European Signal Processing Conference (EUSIPCO 2007)*, Poznan, Poland, Sept. 2007b

Preliminary version: Published as Technical Report LiTH-ISY-R-2821, Dept. of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

Fast particle filters for multi-rate sensors

Thomas B. Schön, David Törnqvist and Fredrik Gustafsson

Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
`{schon, tornqvist, fredrik}@isy.liu.se`

Abstract

Computational complexity is a major concern for practical use of the versatile particle filter (PF) for nonlinear filtering applications. Previous work to mitigate the inherent complexity includes the marginalized particle filter (MPF), with the fastSLAM algorithm as one important case. MPF utilizes a linear Gaussian sub-structure in the problem, where the Kalman filter (KF) can be applied. While this reduces the state dimension in the PF, the present work aims at reducing the sampling rate of the PF. The algorithm is derived for a class of models with linear Gaussian dynamic model and two multi-rate sensors, with different sampling rates, one slow with a nonlinear and/or non-Gaussian measurement relation and one fast with a linear Gaussian measurement relation. For this case, the KF is used to process the information from the fast sensor and the information from the slow sensor is processed using the PF. The problem formulation covers the important special case of fast dynamics and one slow sensor, which appears in many navigation and tracking problems.

1 Introduction

The nonlinear filtering problem deals with estimation of the states x_t, using the information in the measurements up to time t, $y_{1:t} \triangleq \{y_1, \cdots, y_t\}$, for a nonlinear dynamic system,

$$x_{t+1} = f_t(x_t, u_t, w_t), \tag{1a}$$

$$y_t = h_t(x_t, u_t, e_t). \tag{1b}$$

Here, u_t denotes a known input signal, e_t and w_t denote the stochastic measurement and process noise, respectively. Furthermore, the functions f and h contain the dynamic equations for the system and the measurement equations, respectively.

All information about the state that is present in the measurements is contained in the filtering probability density function (PDF) $p(x_t|y_{1:t})$. The particle filter Gordon et al. (1993), Doucet et al. (2000) provides an arbitrary good approximation to this PDF. However, it is a well-known fact from applications that the number of particles that are needed to get a good approximation of the true filtering PDF increases rapidly with the state dimension. Therefore, efficient implementations of the PF have to utilize some kind of structure inherent in the problem. An example of such a structure is when there is a linear Gaussian sub-structure available in (1). This is exploited in the marginalized particle filter (also referred to as the Rao-Blackwellized particle filter Doucet et al. (2000)) by the observation that conditioned on the nonlinear states it is a linear Gaussian system, which can be optimally estimated using the Kalman filter Kalman (1960). The resulting algorithm applies a PF to a low-dimensional part of the state vector, where each particle has an associated Kalman filter estimate of the remaining part of the state vector. For more information on the marginalized particle filter we refer to Schön et al. (2005), Chen and Liu (2000).

In the present contribution we consider a different structure, which arises when there are multi-rate sensors (sensors providing measurements with different sampling times) available. It will be shown that if the inherent structure is exploited by the algorithm the quality of the estimates will be better at a lower computational cost, compared to the direct use of the PF. Algorithmically, there are several similarities between the present contribution and the MPF. We will study a class of filtering problems, where the model (1) has the following structure

$$x_{t+1} = A_t x_t + B_t u_t + G_t w_t, \qquad t = 0, 1, \dots \tag{2a}$$

$$y_{1,t} = C_t x_t + D_t u_t + e_{1,t}, \qquad t = 1, 2, \dots \tag{2b}$$

$$y_{2,t} = h(x_t, u_t, t) + e_{2,t}, \qquad t = r, 2r, \dots. \tag{2c}$$

More specifically,

- The dynamic model is linear with Gaussian process noise $w_t \in \mathcal{N}(0, Q_t)$.
- The fast sensor $y_{1,t}$ is linear with Gaussian measurement noise $e_{1,t} \in \mathcal{N}(0, R_{1,t})$.
- The slow sensor provides measurements $y_{2,t}$ a factor r times slower than the fast sensor, and is a nonlinear function of the state and the input signal, with possibly non-Gaussian measurement noise $e_{2,t} \sim p_{e_{2,t}}(\,\cdot\,)$.

The typical application area we have in mind is navigation, where the fast sensor delivers measurements of the relative movements (inertial measurement unit or odometer) and the slow sensor provides measurements of absolute reference to given landmarks (camera, bearings-only or range sensors). This includes important specific applications, such as Simultaneous Localization and Mapping (SLAM) Thrun et al. (2005), Durrant-Whyte and Bailey (2006) and navigation in sensor networks.

We also want to mention that the special case when the fast sensor is missing, given by (2a, 2c) alone, covers all the navigation and tracking applications surveyed in Gustafsson et al. (2002). The presented algorithm then basically propagates the particles available after the measurement equation to a Gaussian mixture, which is resampled at the time of the next measurement.

2 Nonlinear State Filtering

The solution to the nonlinear state filtering problem is given by the filtering PDF and its conceptual form is given in the subsequent section. Furthermore, a popular approximation to this solution, the particle filter, is briefly introduced in Section 2.2.

2.1 Conceptual Solution

In discussing the conceptual solution to the nonlinear state estimation problem it is convenient to work either with the PDF's or the distribution functions of the involved stochastic variables. In the present work we will work with PDF's. Hence, the first step is to describe model (1) using PDF's. This can be done according to,

$$x_{t+1} \sim p_t(x_{t+1}|x_t), \tag{3a}$$
$$y_t \sim p_t(y_t|x_t), \tag{3b}$$

where $\sim$ denotes distribution according to. Here, $p_t(x_{t+1}|x_t)$ and $p_t(y_t|x_t)$ are commonly referred to as the transition PDF and likelihood, respectively. Note that the deterministic input signal u_t is seen as a part of the model and the conditioning on the model is implicit in the PDF's. More specifically for the model under consideration in this work (2) we have the following transition PDF $p_t(x_{t+1}|x_t)$ and likelihood $p_t(y_t|x_t)$,

$$p_t(x_{t+1}|x_t) = p_{G_t w_t}(x_{t+1} - A_t x_t - B_t u_t), \tag{4a}$$
$$p_t(y_{1,t}|x_t) = p_{e_{1,t}}(y_t - C_t x_t - D_t u_t), \tag{4b}$$
$$p_t(y_{2,t}|x_t) = p_{e_{2,t}}(y_t - h(x_t, u_t, t)), \tag{4c}$$

The solution to the nonlinear state estimation problem using the model representation in (3) follows rather straightforwardly using Bayes' theorem and the Markov property, (see e.g., Jazwinski (1970) for details). The resulting recursions for the filtering and prediction densities are given by

$$p(x_t|y_{1:t}) = \frac{p(y_t|x_t)p(x_t|y_{1:t-1})}{\int p(y_t|x_t)p(x_t|y_{1:t-1})dx_t}, \tag{5a}$$
$$p(x_{t+1}|y_{1:t}) = \int p(x_{t+1}|x_t)p(x_t|y_{1:t})dx_t. \tag{5b}$$

The reason for referring to this as a conceptual solution is that the multidimensional integrals in (5) typically do not allow for an analytical solution. However, there are a few special cases allowing for analytical solutions, such as the linear Gaussian case, when the solution is given by the Kalman filter Kalman (1960). For the more interesting nonlinear and/or non-Gaussian case we are forced to approximations of some kind and the particle filter provides a very interesting and powerful approximation, especially when the problem has an inherent structure which can be exploited.

2.2 Particle Filter

The main idea underlying the particle filter is to approximate the filtering PDF using a finite number of so called particles $\{x_{t|t}^{(i)}\}_{i=1}^{N}$ according to

$$\hat{p}_N(x_t|y_{1:t}) = \sum_{i=1}^{N} \gamma_t \delta(x_t - x_{t|t}^{(i)}), \tag{6}$$

where each particle $x_{t|t}^{(i)}$ has an importance weight γ_t associated to it. Note that $\delta(x_t - x_{t|t}^{(i)})$, denotes the delta-Dirac function located at $x_{t|t}^{(i)}$. The importance weight contains information about how probable the corresponding particle is. Due to the delta-Dirac form used in (6), a finite sum is obtained when this approximation is passed through an integral, i.e., the multidimensional integrals are simplified to finite sums, enabling an approximation to (5). All the details of the particle filter were independently discovered by Gordon et al. (1993), Kitagawa (1996), Isard and Blake (1996). However, the main ideas, save for the crucial resampling step have been around since the 1940's Metropolis and Ulam (1949).

For an introduction to and derivation of the the particle filter we refer to e.g., Arulampalam et al. (2002), Doucet et al. (2000). A rather basic form, used in this contribution, is given in Algorithm 1.

Algorithm 1 Particle filter

1. Initialize the particles, $\{x_{1|0}^{(i)}\}_{i=1}^{N} \sim p(x_0)$.
2. Compute the importance weights $\{\gamma_t^{(i)}\}_{i=1}^{N}$,
   $$\gamma_t^{(i)} = p_t(y_t|x_{t|t-1}^{(i)}), \qquad i = 1, \dots, N,$$
   and normalize $\tilde{\gamma}_t^{(i)} = \gamma_t^{(i)} / \sum_{j=1}^{N} \gamma_t^{(j)}$
3. Draw N new particles with replacement, for each $i = 1, \dots, N$,
   $$Pr(x_{t|t}^{(i)} = x_{t|t-1}^{(j)}) = \tilde{\gamma}_t^{(j)}, \qquad j = 1, \dots, N.$$
4. Predict the particles by drawing independent samples according to
   $$x_{t+1|t}^{(i)} \sim p_t(x_{t+1}|x_{t|t}^{(i)}), \qquad i = 1, \dots, N.$$
5. Set $t := t + 1$ and iterate from step 2.

It is worth stressing that this is the basic form of the particle filter and that there are several embellishments available in the literature. However, for the contribution in this paper there is no reason to depart from the basic form and our ideas can of course be used together with most of the existing particle filters.

3 Multi-rate Particle Filter

Obviously one solution to the problem considered in this paper is simply to neglect the inherent structure and apply the particle filter directly. However, if we choose to make use of the inherent structure we can obtain better estimates at a lower computational cost. In the subsequent section we describe how this can be done and in Section 3.2 an efficient way of computing the estimates is given.

3.1 Algorithm

The algorithm will be explained using an induction type of reasoning. Let us start by assuming that the information in the slow, nonlinear measurement (2c) has just been used. The approximation of the filtering PDF is then given by (6). Now, during the time until the next slow measurement arrives, i.e., for the times $k = t+1, t+2, \dots, t+r$, model (2) is obviously reduced to

$$x_{k+1} = A_k x_k + B_k u_k + G_k w_k, \qquad w_k \sim \mathcal{N}(0, Q_k), \tag{7a}$$

$$y_{1,k} = C_k x_k + D_k u_k + e_{1,k}, \qquad e_{1,k} \sim \mathcal{N}(0, R_k), \tag{7b}$$

which is a linear Gaussian model. This implies, as is well known, that the solution to (5) is available in closed form via the Kalman filter Kalman (1960). Hence, there is no reason to use an approximate method such as the particle filter to compute the estimates for $p(x_k|y_{1,1:k}, y_{2,1:t}), k = t+1, \dots, t+r$, since the Kalman filter will produce the optimal estimate. Furthermore, this can be performed at a lower computational cost using the Kalman filter. The fact that $\hat{p}_N(x_t|y_{1:t})$, given by (6) is non-Gaussian prevents us from direct application of the Kalman filter. However, this can be efficiently handled using parallel Kalman filters, initiated with

$$\hat{x}^{(i)}_{k|k} = x^{(i)}_{k|k}, \qquad i = 1, \dots, N, \tag{8a}$$

$$P^{(i)}_{k|k} = 0, \qquad i = 1, \dots, N. \tag{8b}$$

Note that despite of (8b) the uncertainty is still present, since we run several Kalman filters in parallel. Hence, the uncertainty is inherent in the representation. Furthermore, (8b) is crucial since it implies that we can use the same covariance matrices $P_{k|k}, P_{k+1|k}$ and Kalman gains K_k for all particles and their computations can be performed off-line, once and for all, before the filter is employed. This is important, since it allows us to save computational resources for the on-line computations. The Kalman filtering computations can be performed using the standard recursions, the square-root implementation or any other form. Regardless of which form that is used the particle updates are in the following form

$$x^{(i)}_{t+r|t+r} = g(x^{(i)}_{t|t}, y_{1,t+1:t+r}, u_{t:t+r}), \qquad i = 1, \dots, N, \tag{9}$$

where g denotes a general function. More specifically, if the standard recursions for the Kalman filters are used it is straightforward to verify that the particle updates (9) are given by

$$x^{(i)}_{t+r|t+r} = Lx^{(i)}_{t|t} + \sum_{j=0}^{r} J_j u_{t+j} + \sum_{j=1}^{r} M_j y_{1,t+j}, \tag{10}$$

where $i = 1, \dots, N$ and

$$\begin{aligned}
L &= \prod_{j=r}^{1} (I - K_{t+j}C_{t+j})A_{t+j-1}, && \text{(11a)}\\
J_0 &= \left(\prod_{i=r}^{2} (I - K_{t+i}C_{t+i})A_{t+i-1}\right)(I - K_{t+1}C_{t+i})B_t, && \text{(11b)}\\
J_j &= \left(\prod_{i=r}^{j+2} (I - K_{t+i}C_{t+i})A_{t+i-1}\right)(I - K_{t+j+1}C_{t+j+1})\times \\
&\quad (A_{t+j}K_{t+j}D_{t+j} + B_{t+j}), \quad j = 1, \dots, r-2, && \text{(11c)}\\
J_{r-1} &= (I - K_{t+r}C_{t+r})(A_{t+j}K_{t+j}D_{t+j} + B_{t+j}) && \text{(11d)}\\
J_r &= K_{t+r}, && \text{(11e)}\\
M_j &= \left(\prod_{i=r}^{j+1} (I - K_{t+i}C_{t+i})A_{t+i-1}\right)K_{t+j}, \quad j = 1, \dots, r-1, && \text{(11f)}\\
M_r &= K_{t+r}. && \text{(11g)}
\end{aligned}$$

The covariance matrices and the Kalman gain are given by

$$\begin{aligned}
P_{k+1|k} &= A_k P_{k|k} A_k^T + G_k Q_k G_k^T, && \text{(12a)}\\
K_k &= P_{k|k-1} C_k^T (C_k P_{k|k-1} C_k^T + R_{1,k})^{-1}, && \text{(12b)}\\
P_{k|k} &= P_{k|k-1} - K_k C_k P_{k|k-1}. && \text{(12c)}
\end{aligned}$$

Note that the term $\sum_{j=0}^{r} J_j u_{t+j} + \sum_{j=1}^{r} M_j y_{1,t+j}$ in (10) is the same for all particles, allowing us to save computations.

The final step is at time $t + r$, before the next measurement from the slow sensor delivers its next measurement $y_{2,t+r}$. In order to make use of this nonlinear measurement the result from the Kalman filters has to be assembled into a new set of particles. This is accomplished simply by drawing a set of particles from the current approximation of the filtering PDF $p(x_{t+r}|y_{1,1:t+r}, y_{2,1:t})$,

$$\hat{p}_N(x_{t+r}|y_{1,1:t+r}, y_{2,1:t}) = \sum_{i=1}^{N} \tilde{q}^{(i)}_{t+r} \mathcal{N}(x_{t+r}|x^{(i)}_{t+r|t+r}, P_{t+r|t+r}). \tag{13}$$

and the induction is complete. Note that the weights $q^{(i)}_{t+r}$ are given by (see Orguner (2006) for details)

$$\tilde{q}^{(i)}_{t+r} = \frac{\prod_{k=0}^{r} p(y_{1,t+k}; \hat{y}^{(i)}_{1,t+k}, S_{t+k})}{\sum_{j=1}^{N} \prod_{k=0}^{r} p(y_{1,t+k}; \hat{y}^{(i)}_{1,t+k}, S_{t+k})}. \tag{14}$$

Algorithm 2 Multi-rate particle filter

1. Initialize the particles, $\{x_{1|0}^{(i)}\}_{i=1}^{N} \sim p(x_0)$.
2. Compute the importance weights $\{\gamma_t^{(i)}\}_{i=1}^{N}$

$$\gamma_t^{(i)} = p_t(y_{2,t}|x_{t|t-1}^{(i)}), \qquad i = 1, \dots, N, \tag{15}$$

   and normalize $\tilde{\gamma}_t^{(i)} = \gamma_t^{(i)} / \sum_{j=1}^{N} \gamma_t^{(j)}$.

3. Draw N new particles with replacement, for each $i = 1, \dots, N$,

$$Pr(x_{t|t}^{(i)} = x_{t|t-1}^{(j)}) = \tilde{\gamma}_t^{(j)}, \qquad j = 1, \dots, N. \tag{16}$$

4. Update the particles using the information in $\{y_{1,t+i}\}_{i=1}^{r}$ and $\{u_{t+i}\}_{i=1}^{r}$according to (10).
5. Draw N new particles,

$$x_{t+r|t+r}^{(i)} \sim \hat{p}_N(x_{t+r}|y_{1,1:t+r}, y_{2,1:t}), \quad i = 1, \dots, N,$$

   where $\hat{p}_N(x_{t+r}|y_{1,1:t+r}, y_{2,1:t})$ is given by (13).

6. Set $t := t + r$ and iterate from step 2.

To sum up, we have now motivated Algorithm 2 below.

An obvious extension is the combined use of the marginalized particle filter and the ideas presented in this paper. The application of such an algorithm to the SLAM problem for a robot moving in 3D equipped with vision and inertial sensors could probably result in interesting results, see Schön et al. (2007). Note that this might require linearization of certain equations to get a model in the form (2).

3.2 Computing Estimates

The, in many respects, optimal estimate for the state at time t is the conditional expectation,

$$\hat{x}_{t|t} = \mathsf{E}(x_t|y_{1:t}). \tag{17}$$

Using the standard particle filter (Algorithm 1) this estimate is computed after step 2 and it is given by

$$\hat{x}_{t|t} = \sum_{i=1}^{N} \tilde{\gamma}_t x_{t|t}^{(i)}. \tag{18}$$

When the slow measurement $y_{2,t}$ has just been used in step 2 in the multi-rate particle filter (Algorithm 2) the conditional mean estimate (17) is of course given by (18) as well. The intermediate estimates, between two slow measurements can be computed in a similar fashion. However, that would be very computationally intensive. Using the fact that the order of linear operations can be interchanged will save a lot of computations here. To be more specific,

$$\hat{x}_{t+l|t+l} = \sum_{i=1}^{N} \tilde{q}_{t+l}^{(i)} x_{t+l|t+l}^{(i)} = \sum_{i=1}^{N} (L_l \tilde{q}_{t+l}^{(i)} x_{t|t}^{(i)} + M_l) \tag{19}$$

$$= L_l \sum_{i=1}^{N} \tilde{q}_{t+l}^{(i)} x_{t|t}^{(i)} + M_l, \tag{20}$$

where

$$L_l = \prod_{j=l}^{1} (I - K_{t+j} C_{t+j}) A_{t+j-1}, \tag{21}$$

Hence, the intermediate estimates can be computed at almost no cost at all.

4 Simulations

In this section, the multi-rate particle filter (Algorithm 2) will be compared to the standard particle filter (Algorithm 1). This is done using Monte Carlo simulations for an illustrative example.

4.1 Setup

Consider a vehicle moving in a two dimensional world. Information about this vehicle is obtained using two multi-rate sensors. One sensor measures the range to a fixed object at a known position (the origin in this case) at 1 Hz and the other sensor measures the velocity $(v_{x,t},\ v_{y,t})$ at 10 Hz. For convenience we assume that the vehicle can be modelled using a constant velocity model according to,

$$\begin{pmatrix} p_{x,t+1} \\ p_{y,t+1} \\ v_{x,t+1} \\ v_{y,t+1} \end{pmatrix} = \underbrace{\begin{pmatrix} 1 & 0 & T_s & 0 \\ 0 & 1 & 0 & T_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{A} \begin{pmatrix} p_{x,t} \\ p_{y,t} \\ v_{x,t} \\ v_{y,t} \end{pmatrix} + \underbrace{\begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}}_{G} \begin{pmatrix} w_{1,t} \\ w_{2,t} \end{pmatrix}, \tag{22a}$$

$$y_{1,t} = \underbrace{\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{C} \begin{pmatrix} p_{x,t} \\ p_{y,t} \\ v_{x,t} \\ v_{y,t} \end{pmatrix} + e_{1,t}, \tag{22b}$$

$$y_{2,t} = \underbrace{\sqrt{p_{x,t}^2 + p_{y,t}^2}}_{h(x_t)} + e_{2,t}, \tag{22c}$$

where $(p_{x,t},\ p_{y,t})$ and $(v_{x,t},\ v_{y,t})$ denote the position and velocity, respectively. Hence, the model is in the form (2) and the multi-rate particle filter given in Algorithm 2 can be applied. Model (22) is time-invariant and there are no input signals present. Hence, (10) – (11) can be simplified according to

$$x_{t+r|t+r}^{(i)} = Lx_{t|t}^{(i)} + \sum_{j=1}^{r} M_j y_{1,t+j}, \tag{23}$$

where $i = 1, \dots, N$ and

$$L = \prod_{j=r}^{1} (I - K_{t+j}C)A, \tag{24a}$$

$$M_j = \left(\prod_{l=r}^{j+1} (I - K_{t+l}C)A \right) K_{t+j}, \quad j = 1, \dots, r-1, \tag{24b}$$

$$M_r = K_{t+r}. \tag{24c}$$

Furthermore, the sampling time is set to be $T_s = 1/10\,\text{s}$, the covariance for the process noise is set to be $\text{Cov}\left((w_{1,t}\ w_{2,t})^T\right) = 0.5I_2$ and the covariance for the measurement noise is $\text{Cov}\left((e_{1,t}^T\ e_{2,t})^T\right) = I_3$. We have used 1000 Monte Carlo simulations in order to obtain reliable results.

4.2 Simulation Results

The root mean square errors (RMSE) over the Monte Carlo simulations for the position and velocity estimates are shown in Figure 1. The error in position estimates is lower for the multi-rate particle filter, whereas the velocity errors are almost identical. Note that the sawtooth pattern in the position error is due to a rapidly improved estimate every time an update on the range is available. The increasing error over time is due to incomplete observability. Measuring the range and velocity is not enough to estimate the position without drift.

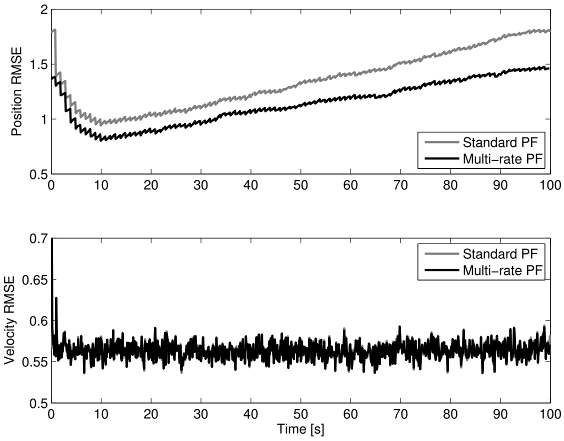


Figure 1: *RMSE in position (top) and velocity (bottom) for* 1000 *Monte Carlo simulations, using* 3000 *particles.*

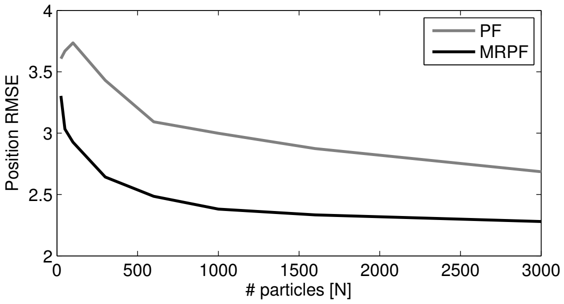


Figure 2: *Position RMSE as a function of the number of particles, using* 1000 *Monte Carlo simulations.*

In Figure 2 the RMSE for the position is provided as a function of the number of particles used in the filters. Here, we can see that the multi-rate particle filter performs better, but the difference becomes smaller as the number of particles increases. This is expected and in accordance to theory, since the particle filter converges to the optimal filter as the number of particles tends to infinity.

The computational time using the multi-rate particle filter is decreased, especially when a large number of particles are used. This is illustrated in Figure 3.

In studying particle filters it is always interesting to study the rate of divergence as

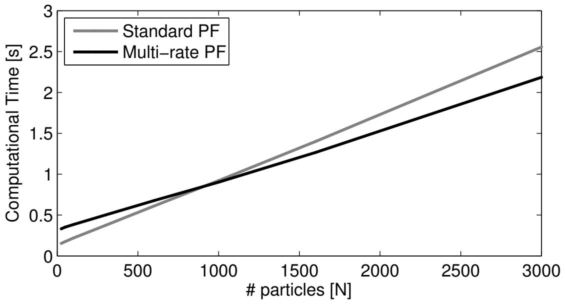


Figure 3: *Illustration of the computational time as a function of the number of particles. The multi-rate particle filter requires fewer computations for large number of particles, compared to the standard particle filter.*

a function of the number of particles used in the filter. First of all, let us define what is meant by divergence in this context. The filter is said to diverge whenever the sum of the importance weights fall below a certain problem dependent threshold $\lambda_t > 0$, i.e., when

$$\sum_{i=1}^{N} p(y_{2,t}|x_{t|t-1}^{(i)}) = \sum_{i=1}^{N} \gamma_t^{(i)} < \lambda_t. \tag{25}$$

The motivation for this choice of divergence test is simply that it indicates when the particle cloud is too far from what is indicated by the present measurement. This is more likely to happen if there are fewer particles. Figure 4 shows the rate of divergence as a function of the number of particles. The performance of the multi-rate particle filter is significantly better for small N and slightly worse for large N. This makes sense since there is no resampling during the Kalman updates, effectively giving higher variance to the weights. To redeem this, resampling can be performed more often at the cost of computational speed.

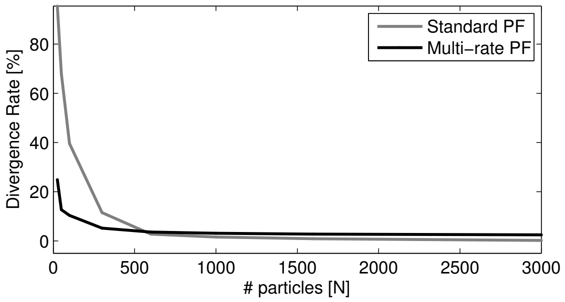


Figure 4: *Rate of divergence as a function of the number of particles used in the filters.*

5 Conclusion

We have proposed a new algorithm for state estimation, which can be used when the underlying dynamical model has a certain structure. The algorithm is based on the particle filter and exploits a linear Gaussian sub-structure. The resulting algorithm produces estimates of better quality at a lower computational cost, when compared to the standard particle filter. Finally, the structure commonly arises in sensor fusion applications, when there is one slow (nonlinear) and one fast (linear) sensor available.

References

M. Arulampalam, S. Maskel, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, Feb. 2002.

R. Chen and J. S. Liu. Mixture Kalman filters. *Journal of the Royal Statistical Society*, 62(3):493–508, 2000.

A. Doucet, S. J. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping (SLAM): Part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110, June 2006.

N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings on Radar and Signal Processing*, volume 140, pages 107–113, 1993.

F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P.-J. Nordlund. Particle filters for positioning, navigation and tracking. *IEEE Transactions on Signal Processing*, 50(2):425–437, Feb. 2002.

M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proceedings of the European Conference on Computer Vision*, volume 1, pages 343–356, Cambridge, UK, 1996.

A. H. Jazwinski. *Stochastic processes and filtering theory*. Mathematics in science and engineering. Academic Press, Inc, New York, USA, 1970.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the American Society of Mechanical Engineering — Journal Basic Engieering*, 82 (Series D):35–45, Mar. 1960.

G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996.

N. Metropolis and S. Ulam. The Monte Carlo method. *Journal of the American Statistical Association*, 44(247):335–341, 1949.

U. Orguner. *Improved state estimation for jump Markov linear systems.* PhD thesis, Middle east technical university, Ankara, Turkey, 2006.

T. Schön, F. Gustafsson, and P.-J. Nordlund. Marginalized particle filters for mixed linear/nonlinear state-space models. *IEEE Transactions on Signal Processing*, 53(7): 2279–2289, July 2005.

T. Schön, R. Karlsson, D. Törnqvist, and F. Gustafsson. A framework for simultaneous localization and mapping utilizing model structure. In *The 10th International Conference on Information Fusion*, Quebec, Canada, Aug. 2007.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics.* The MIT Press, 2005.

Paper E

A multiple UAV system for vision-based search and localization

Authors: John Tisdale, Allison Ryan, Zu Kim, David Törnqvist, and J. Karl Hedrick

Edited version of the paper: J. Tisdale, A. Ryan, Z. Kim, D. Törnqvist, and J. K. Hedrick. A multiple UAV system for vision-based search and localization. In *American Control Conference*, Seattle, WA, USA, June 2008

Preliminary version: Published as Technical Report LiTH-ISY-R-2865, Dept. of Electrical Engineering, Linköping University, SE-581 83 Linköping, Sweden.

A multiple UAV system for vision-based search and localization

John Tisdale*, Allison Ryan*, Zu Kim*, David Törnqvist**, and J. Karl Hedrick*

*Dept. of Mechanical Engineering
University of California, Berkeley
Berkeley, California 94720, USA
`{jtisdale, allisonr, zuwhan}@berkeley.edu`,
`khedrick@me.berkeley.edu`

**Dept. of Electrical Engineering,
Linköping University,
SE–581 83 Linköping, Sweden
`tornqvist@isy.liu.se`

Abstract

The contribution of this paper is an experimentally verified real-time algorithm for combined probabilistic search and track using multiple unmanned aerial vehicles (UAVs). Distributed data fusion provides a framework for multiple sensors to search for a target and accurately estimate its position. Vision based sensing is employed, using fixed downward-looking cameras. These sensors are modeled to include vehicle state uncertainty and produce an estimate update regardless of whether the target is detected in the frame or not. This allows for a single framework for searching or tracking, and requires non-linear representations of the target position probability density function (PDF) and the sensor model. While a grid-based system for Bayesian estimation was used for the flight demonstrations, the use of a particle filter solution has also been examined.

Multi-aircraft flight experiments demonstrate vision-based localization of a stationary target with estimated error covariance on the order of meters. This capability for real-time distributed estimation will be a necessary component for future research in information-theoretic control.

1 Introduction

Unmanned Aerial Vehicles (UAVs) have proven successful in information gathering tasks, such as target search or environmental sensing. To robustly estimate a target state, an information gathering system must incorporate models for uncertainty in the sensor, target model and a model for its own state. Robust estimation becomes more important as teams of agents use these estimates to plan their own motion, as in Hoffmann et al. (2006), Grocholsky (2005), Mathews and Durrant-Whyte (2007). In the case where an agent has noisy sensors, or a limited sensor field of view, the use of probabilistic estimation becomes more important to develop accurate target estimates.

Traditionally, probabilistic ‘search’ problems represent the unknown state of a target by decomposing the search space into as many regions as is computationally feasible and updating each region as it is observed. ‘Track’ problems generally represent target state as a multivariate parameterized PDF (often Gaussian), but the choice of representation precludes incorporation of sensor readings that do not detect the target. In this work, we present a unified framework that accounts for ‘search’ by updating the target probability density function even when the target is not detected, but also achieves localization or tracking by updating an estimate using multiple observations of the target.

Campbell *et. al.* Campbell and Wheeler (2006) demonstrate a UAV based system for geolocation; a sigma point filter is used to filter sensor data from a gimbaled camera to estimate target location. Low-level target tracking guarantees that the target will be observed at all times. The work presented here differs from the work of Campbell and Wheeler (2006) in that the search for targets is addressed in addition to target localization. Additionally, this work uses a decentralized multi-sensor approach, in which sensor readings are fused across a team of vehicles.

Probabilistic maps for search have been used extensively; recently the authors in Vidal et al. (2002) used probabilistic maps for pursuit evasion games using unmanned helicopters. While search was explicitly addressed, localization was not examined in the framework of this paper.

The development of a unified framework for search and track was first proposed in Furukawa et al. (2006). Furukawa *et. al.* use a grid-based PDF to represent target position in the plane; this approach is effective for search, but falls short computationally once the target has been effectively localized. The precision of the estimate is limited by the resolution of the grid, and states with more than two dimensions (such as position and velocity in two dimensions) are difficult to model. In this work, a particle filter is used, in addition to a grid, in the hope that better localization results may be obtained in a search and localization framework, with similar computational effort.

The major contributions of this paper are the development and demonstration of a UAV based system for decentralized Bayesian search and localization. A grid-based Bayesian filter and a particle filter are used to maintain a target estimate. This system uses off the shelf hardware components to search for targets, and once detected, localize their position. High localization accuracy is achieved using low-cost UAVs by modeling the sensor uncertainty and fusing multiple observations using a Bayesian estimator.

2 Decentralized Filtering for Search and Track

A decentralized estimator provides a method to update an estimate using multiple observations and a process model of the target state evolution. Recursive Bayes estimation is a method to update the filtering density $p(x_k|z_{1:k})$ using a sensor model $p(z_k|x_k)$, and a process model $p(x_{k+1}|x_k)$. The observation z_k is applied in the update step (1a) and the process model is applied in the prediction step (1b).

$$p(x_k|z_{1:k}) = \frac{p(z_k|x_k)p(x_k|z_{1:k-1})}{p(z_k|z_{1:k-1})} \tag{1a}$$

$$p(x_{k+1}|z_{1:k}) = \int p(x_{k+1}|x_k)p(x_k|z_{1:k})dx_k \tag{1b}$$

The sensor fusion problem is also easily solved in this framework using the sensor model $p(z_k|x_k)$ to relate the sensors. In our problem, the measurements (z_k^1 and z_k^2) from the two airplanes are independent so

$$p(z_k|x_k) = p(z_k^1, z_k^2|x_k) = p(z_k^1|x_k)p(z_k^2|x_k). \tag{2}$$

In the case where our distributions are Gaussian and the models are linear, (1) can be solved analytically and results in the well-known Kalman Filter Kalman (1960). For almost all other cases, analytical solutions cannot be obtained and approximations have to be used. The most common approximation is the Extended Kalman Filter (EKF) Kailath et al. (2000), that linearizes the system around the current state estimate and assume Gaussian distributions. To treat more complex distributions, the Unscented Kalman Filter (UKF) can be used Julier (2002). This filter uses sigma points to represent the distribution undergoing a non-linear transform. While this approach has been used successfully in Campbell and Wheeler (2006), Frew (2006), the EKF and UKF are unable to model highly multi-modal or skewed distributions. Other methods for treating arbitrary distributions with non-linear models are approximating the distributions spatially as a grid or with movable gridpoints (particles). In the grid-case, (1) is computed for each grid point. The case with particles result in the particle filter, see Gordon et al. (1993).

In this paper, where a search task is treated, the distributions are far from Gaussian. A grid-based or particle filter approach are therefore more suited, since they better captures the successive application of 'no detection' updates on a prior target distribution that might be uniform over a wide area.

Numerous approaches to data fusion have been explored by Durrant-Whyte *et. al.* Grocholsky et al. (2000, 2003). These techniques, such as the information filter and channel filter, often exploit the filtering technique to minimize the information communicated between nodes. In the work presented here, a parameterization of the likelihood function $p(z_k|x_k)$ is available and the parameters are distributed among the aircraft. The estimate is then updated using (1a). The likelihood function is given by one of two functional forms for the vision sensor: the case in which the target is observed at some camera coordinates and the case where it is not observed.

3 Modeling and Implementation

An accurate model that captures the inherent uncertainty of an imperfect sensor is a crucial component of any real-world estimation system. Including observations in which the target is not detected enables probabilistic search and localization using all available information and requires sensor models for both the 'detect' (D) and 'no detect' ($\bar{D}$) cases. The 'detect' sensor model $p(D_k, z_k|x_k)$ is used to update the estimate when the target

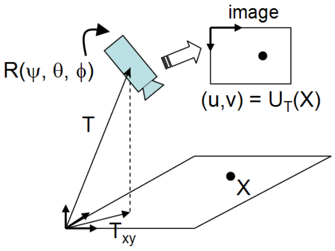


Figure 1: *Coordinate systems. There is a fixed coordinate system and an image coordinate system, the rotation between them is described by the Euler angles* ψ, θ, ϕ.

is detected at image coordinates $z_k = (u, v)$. The 'no detect' sensor model $p(\bar{D}_k|x_k)$ provides the update when the target is not detected in the frame. The time subscript will be dropped, assuming that all values correspond to the present time.

The sensor is modeled as a standard pinhole camera and the probability of detecting a target in the field of view is based on the image resolution at the target location. The visibility of a target in the field of view depends on the aircraft position T and orientation defined by the Euler angle vector $q = [\psi\ \theta\ \phi]$. When the aircraft state is unknown, the visibility is a random variable which can be conditioned on the aircraft state estimates. We derive a probability of true detection, $p(D, V|x)$, and assume that the probability of false detection is constant per frame, $p(D, \bar{V}|x) = P_{false}$. We also assume that the location of a false detection is uniformly distributed over the image area

$$\begin{aligned} p(D, z|x) &= p(z|x, D, V)p(D, V|x) + p(z|x, D, \bar{V})p(D, \bar{V}|x) \\ &= p(z|x, D, V)p(D, V|x) + (\text{image area})^{-1}P_{false}. \end{aligned} \tag{3}$$

The detection likelihood $p(D, V|x)$ and localization likelihood $p(z|x, D, V)$ components of the model are derived separately.

3.1 Target Localization Model

The aircraft translation $T = [T_{XY}\ h]$ and Euler angle vector q define a transformation $U_T(x, T, q) : \mathcal{R}^2 \rightarrow \mathcal{R}^2$ from ground to image coordinates, as shown in Fig. 1. This transformation also depends on the intrinsic camera parameters such as focal length, which are estimated using standard calibration procedures, see e.g., Horn (2000). If T and q are known, we expect to observe the target at $z = U_T(x, T, q)$, and so $p(z|x, D, V) = \delta_z(z - U_T(x, T, q))$, where δ_z is a two-dimensional delta function in image coordinates.

When the aircraft state is unknown, $p(z|x, D, V)$ requires marginalizing over state estimates (4). We assume that each of the Euler angles, altitude, and ground position are estimated independently, breaking down $p(T, q)$ by individual state estimates.

$$p(z|x,D,V) = \int\limits_{T,q} p(z|x,D)p(T,q) = \int\limits_{T,q} \delta_z(z - U_T(x,T,q))p(T,q)$$
$$= \int \delta_z(z - U_T(x,T,q))p(T_{XY})p(h)p(\psi)p(\theta)p(\phi)\, dT_{XY}\, dh\, d\psi\, d\theta\, d\phi \quad (4)$$

For small estimation errors, the transformation $U_T(x,T,q)$ can be approximated by a first order Taylor expansion about the state estimate, so that $P(z|x,D,V)$ for uncertain aircraft state becomes a series of one-dimensional convolutions. The partial derivatives of $U_T(x,T,q)$ with respect to translation are simply components of the rotation matrix, and derivatives with respect to the Euler angles are obtained by differentiating the rotation matrix.

When the UAV state estimation errors can be approximated as Gaussian or Gaussian mixture models (GMMs), the PDF $p(z|x,D,V)$ will be Gaussian (or GMM) in image coordinates z because the set of Gaussians is closed under convolution. However, the likelihood function on ground coordinates x will not be Gaussian or GMM due to the non-linear dependance on the Euler angles. This is similar to the effect seen in range-bearing sensors, where the likelihood function has a characteristic banana-like shape due to the bearing angle estimate uncertainty.

3.2 Target Detection Model

The probability of a true detection $p(D,V|x)$ is a function of the image resolution $r(z,T,q)$ at the target's image coordinates, $z = U_T(x,T,q)$, as well as the visibility of the target. The product of four step functions $s_i(z)$ along the edges of the image defines the visible region in image coordinates. For uncertain aircraft state, marginalization is required as in the previous section.

$$p(D,V|x) = \int\limits_{T,q} p(D|V,x,T,q)p(V|x,T,q)p(T,q)$$
$$= \int\limits_{T,q} p(D|V,r(z,T,q)) \prod_{i=1}^{4} s_i(z)p(T,q) \quad (5)$$

The dependence on resolution can be fitted to experimental data including true and missed detections using Bayes' rule (6). $p(D)$ is the detection rate for the data set and $p(r(z,T,q)|D)$ and $p(r(z,T,q)|\bar{D})$ can each be fitted as a parametric distribution (in this case Gaussian) from the data set.

$$p(D|r(z,T,q)) = \frac{p(r(z,T,q)|D)p(D)}{p(r(z,T,q)|D)p(D) + p(r(z,T,q)|\bar{D})p(\bar{D})} \quad (6)$$

With the resolution and camera coverage expressed in image coordinates, the aircraft state uncertainties will be modeled independently as in the previous section and represented as a two-dimensional convolution mask in image coordinates. Therefore the

Table 1: *Detection modeling based on calibration data set.*

$p(res\|D)$	$\mathcal{N}(60, 40.06)$
$p(res\|\bar{D})$	$\mathcal{N}(60, 62.03)$
$P(D\|V)$	0.96
P_{false}	0.004

marginalization in (5) corresponds to application of the convolution mask $M(\hat{T}, \hat{q})$. We convolve the aircraft state uncertainty with the nominal likelihood function $p(D|x; \hat{T}, \hat{q})$ based on the estimated state $(\hat{T}, \hat{q})$. Recall that detection likelihood function $p(D|x)$ includes both true and false detection.

$$\begin{aligned} p(D|x) &= p(D, V|x) + p(D, \bar{V}|x) \\ &= p(D|V, r(z, \hat{T}, \hat{q})) \prod_{i=1}^{4} s_i(z) * M(\hat{T}, \hat{q}) + P_{false} \end{aligned} \tag{7}$$

For Gaussian state estimates, the form of the detection likelihood is convolution of a Gaussian and a step function in two dimensions, resulting in cumulative Gaussian distributions. As these have no closed form equation, Sigmoid functions are used to approximate the image boundaries, where the Sigmoid function parameters may vary for best fit along each edge.

Fig. 2 shows an example no-detect likelihood function used in the flight experiments. Dark areas of the plot indicate values close to 1, whereas the lighter areas indicate values closer to zero. The effect of decreasing camera resolution may be seen in the upper right side of the field of view. The camera and aircraft parameters for this example are shown in Tables 1 and 2, respectively. Representations such as a particle filter or grid based filter are better suited than EKF methods for this type of likelihood, which is obviously neither Gaussian or convex.

3.3 Target Model

In this work, it is assumed that only a single target is in the area of interest. Multiple targets have been consider in Wong et al. (2005). The primary difficulty in dealing with multiple targets is dealing with the problem of data association, or determining which sensor measurement comes from which target. In this work, we will not examine the data association problem, as there is already a rich literature on the topic. Also, the target is assumed to be static; no update step is necessary.

4 Experimental Results and Analysis

Flight tests of a two-aircraft team with fixed downward-looking cameras were performed at Camp Roberts, CA in August of 2007. The airframe and infrastructure have been previously described in Ryan et al. (2006), Tisdale et al. (2006). A series of flight tests

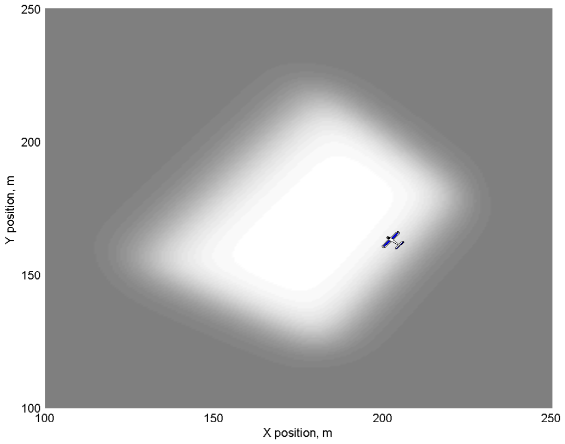


Figure 2: *An example of a no-detect likelihood function. Dark areas indicate values close to 1, whereas the lighter areas indicate values closer to zero.*

were performed to verify sensor modeling, localization accuracy and system performance. Both grid-based Bayesian estimation and particle filtering were tested as representations for the estimated target state.

The grid representations were tested in flight, to update the target position in real-time. Real-time target position data was relayed over an 802.11b wireless link to a ground-based visualization GUI. The particle filter representation was tested post-flight, on the same data collected from the grid-based flight; while the particle filter was not run in real time, testing indicates that is fast enough to do so. Video was processed at roughly four frames per second.

For the sake of real-time data fusion, likelihoods were distributed between aircraft via an ad-hoc wireless network. Flight tests indicated that the wireless network was sufficient to ensure that all likelihoods were communicated between aircraft.

4.1 Grid-based Estimation

Fig. 5 shows the evolution of the searching process with dark areas representing high probability. Snapshots of the search density function and aircraft trajectories are shown during the 'search' portion of the mission, at 70 and 105 seconds, before the target has been detected. UAV 2 detects and localizes the target at roughly 120 seconds. For this flight experiment, the prior was uniform over a one kilometer by one kilometer square. Each grid cell has a width of 12.5 meters. In this flight, the system effectively localized the target to within the resolution of the filter; a red star is used to indicate target position when the target has been localized to a single grid cell. Clearly, the granularity of the grid is a limitation on the possible localization accuracy.

For the purposes of path planning for target search, Bourgault et al. (2003) demonstrated that cumulative probability of detection, $p(D)$ is an appropriate metric for determining the a priori value of a search trajectory. When the target is not detected over a

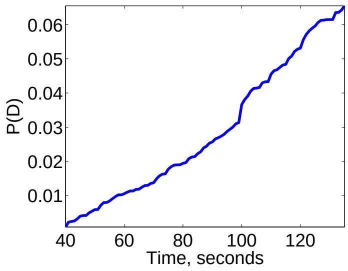


Figure 3: *Cumulative Probability of Detection.*

series of k steps, the probability of detection may be found by (8). Fig. 3 shows the cumulative probability of detection.

$$P(D) = 1 - \int p(x_k | z_{1:k} = \bar{D})\, dx_k = 1 - \int p(x_k) \prod_{j=1}^{k} p(z_j = \bar{D} | X)\, dx_k \tag{8}$$

Fig. 4 shows post processing of the same flight data using a grid cell size of 0.67 meters. Four consecutive filtering PDFs are shown, beginning with the first observation of the target. Coverage of the entire search area at this grid density would require over two million cells, which is not feasible for real time performance on the current flight computer. However, this post processing provides an idea of the system performance that could be achieved with more computing power using the same sensors. Because the quantization error for the fine grid is greatly reduced, this implementation can be used as a benchmark for comparing the accuracy of the original grid and particle filter implementations.

4.2 Particle Filter Implementation

The particle filter used is the standard Sampling Importance Resampling (SIR) particle filter first introduced by Gordon et al. (1993). For a more recent overview, see Arulampalam et al. (2002). The SIR version of the particle filter resamples the particles after each time step, maintaining an equal weight for all particles. This makes the probability proportional to the density of particles in a certain area. The number of particles used is 6400, which is equal to the number of cells in the one square kilometer grid.

The initial probability is again uniform as in the previous section. The target position is modeled as a random walk according to

$$x_{k+1} = x_k + w_k, \tag{9}$$

with $\mathrm{Cov}(w_k) = \mathrm{diag}(1,\, 1)$ because the resampling step requires process noise. Fig. 5(b) shows the target location estimate using a particle filter.

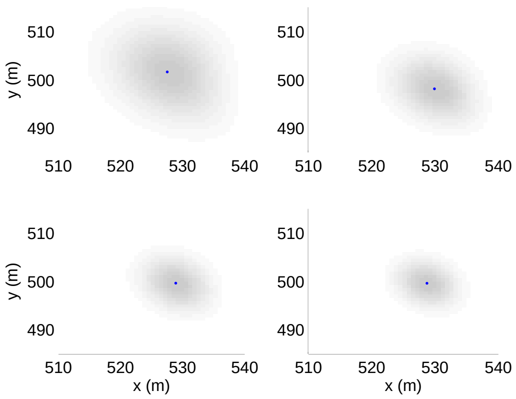
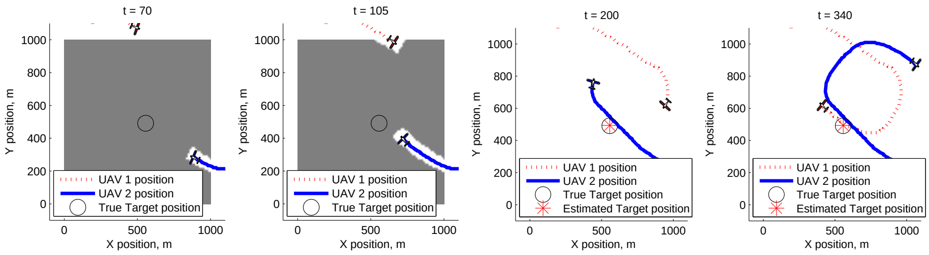
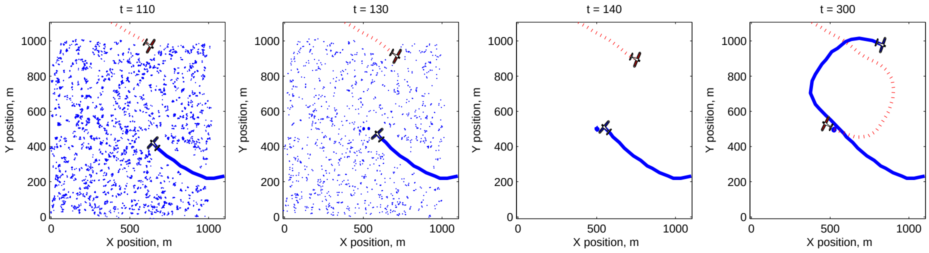


Figure 4: *Evolution of filtering PDF over four consecutive observations, beginning with the first observation of the target. Dark area represents high probability, and the mean of each PDF is marked.*



(a) *Grid-based filter*



(b) *Particle Filter*

Figure 5: *(a) Evolution of the grid-based PDF throughout the search and localization process, with high probability shown as dark (b) Evolution of the particle filter throughout the search and localization process.*

4.3 Error Analysis and Comparison

The implemented grid cell size of 12.5 meters limits the localization accuracy through quantization effects, and cannot be significantly reduced using the current flight computer. The interpretation of the particle filter as an adaptive grid suggests that bias in the estimate due to quantization can be reduced without increasing computation cost by moving particles to represent areas of interest in greater detail. This is demonstrated in Fig. 5, where the particles concentrate around the observed location of the target, and in Fig. 6 where the particle filter produces an estimate which closely corresponds to that of the post-processed high-resolution grid. Covariance ellipses corresponding to the filtering PDFs also show the effect of a large grid size.

Fig. 7 shows time evolution of a scalar measure of covariance for the three implementations. Again considering the fine grid as a benchmark, we observe that the estimate uncertainty decreases with the observations from each sensor and otherwise remains constant for the stationary target. The initial covariance for the fine grid represents a reduced search area. After the target is observed, the square kilometer grid always has higher uncertainty due to quantization. The covariance of the particle filter becomes very small when the target is observed, but then increases due to the required artificial process noise. This increased uncertainty is an artifact of the filter implementation and is a drawback of particle filtering of a stationary target. The uncertainty ellipse shown in Fig. 6 represents the minimum particle filter uncertainty, which is achieved immediately after the final observation.

In addition to quantization, aircraft state estimate uncertainty also contributes to the estimation error. To analyze the contribution of aircraft state uncertainty, we assume that the camera is aligned with the aircraft $\mathbf{e}_3$ (local down) axis and the target appears in the center of the image. Therefore, the target position $\mathbf{x}$ lies along the $\mathbf{e}_3$ direction with $\mathbf{x} \cdot \mathbf{e}_3 = 0$, where $\{\mathbf{e}_i\}$ are fixed world coordinates. The partial derivatives of the resulting expression (10) for $\mathbf{x}$ result in the target position estimate uncertainties shown in Table 2, based on linearization at a set of nominal parameter values.

The largest contributor to target estimate uncertainty is the UAV's position along the direction of motion, which has high uncertainty due to 4 Hz GPS updates. This uncertainty will be reduced in future work by filtering the GPS using angular data. Pitch and roll angle also contribute significantly, especially for larger nominal angles or low quality INS sensors.

$$\begin{aligned}\mathbf{x} = \hat{T}_X\mathbf{e}_1 + \hat{T}_Y\mathbf{e}_2 + \frac{h}{\cos\phi\cos\theta}(\cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi)\mathbf{e}_1 + \\ \frac{h}{\cos\phi\cos\theta}(\cos\phi\sin\theta\sin\psi - \sin\phi\cos\psi)\mathbf{e}_2 \end{aligned} \tag{10}$$

5 Conclusions

This work presents a decentralized approach to target search and localization using low-cost UAVs and sensors. Using low-resolution fixed cameras, a team of UAVs can successfully search a significant area and localize a target with high accuracy by filtering

Table 2: *Error modeling for localization*

Parameter	nominal	σ	σ localization (m)
T_X (along-track)	n/a	10.5 m	10.5
T_Y (cross-track)	n/a	5.5 m	5.5
h	90 m	5.5 m	1.77
ψ	0 r	0.01 r	0.29
θ	0.17 r	0.05 r	4.65
ϕ	0.26 r	0.02 r	1.96

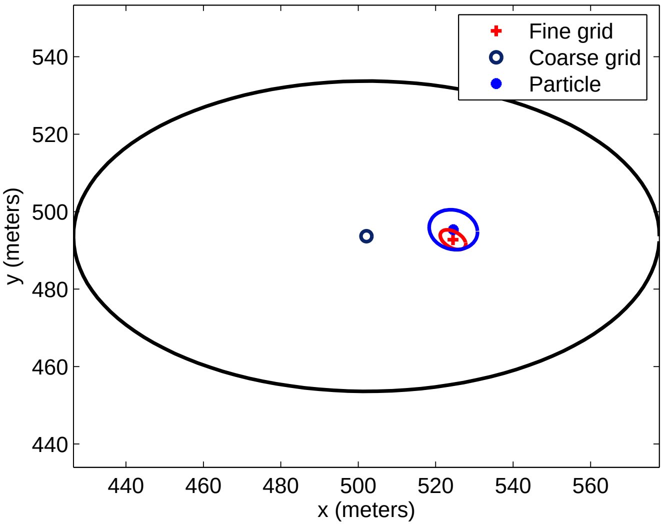


Figure 6: *Mean and covariance ellipse (sigma squared) for fine grid, coarse grid, and particle filter final estimates. Note that the particle filter representation achieves smaller bias error with computation cost similar to the coarse grid.*

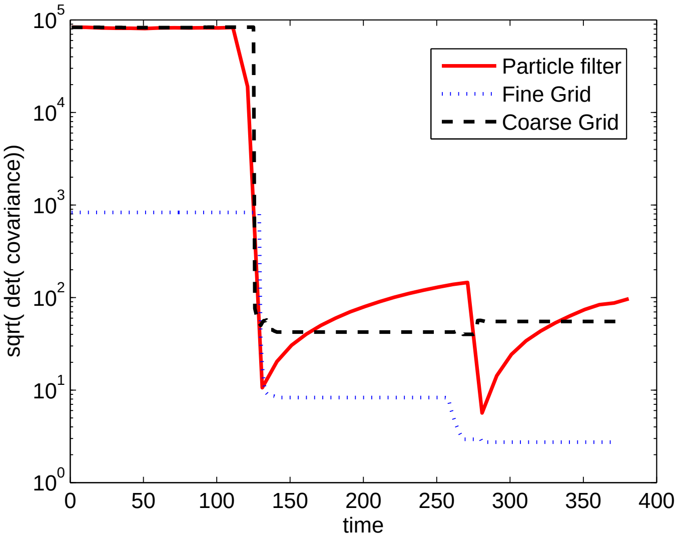


Figure 7: *Time evolution of covariance for fine grid, coarse grid, and particle filter. The particle filter covariance increases over time, due to a random walk target model.*

multiple observations. This work has demonstrated the importance of choosing an appropriate representation for the target PDF. The grid-based distribution is effective for the search problem, the particle filter approach was shown to produce superior localization under the same computational constraints.

References

M. Arulampalam, S. Maskel, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, Feb. 2002.

F. Bourgault, T. Furukawa, and H. Durrant-Whyte. Optimal search for a lost target in a bayesian world. In *Proceedings of the International Conference on Field and Service Robotics*, 2003.

M. E. Campbell and M. Wheeler. A vision based geolocation tracking system for UAVs. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Aug. 2006.

E. Frew. Receding horizon control under uncertainty using optimal input design and the unscented transform. In *IEEE Conference on Decision and Control*, San Diego, CA, Dec. 2006.

T. Furukawa, F. Bourgault, B. Lavis, and H. Durrant-Whyte. Recursive bayesian search-

and-tracking using coordinated UAVs for lost targets. In *Proceedings of the IEEE international conference on robotics and automation*, Orlando, Florida, May 2006.

N. J. Gordon, D. J. Salmond, and A. F. M. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings on Radar and Signal Processing*, volume 140, pages 107–113, 1993.

B. Grocholsky. Information driven coordinated air-ground proactive sensing. In *Proceedings of the IEEE international conference on robotics and automation*, Apr. 2005.

B. Grocholsky, A. Makarenko, T. Kaupp, and H. F. Durrant-Whyte. Scalable control of decentralised sensor platforms. In *2nd International Workshop on Information Processing in Sensor Networks*, pages 96–112, 2003.

B. P. Grocholsky, H. F. Durrant-Whyte, and P. W. Gibbens. Information-theoretic approach to decentralized control of multiple autonomous flight vehicles. In *Proc. SPIE (International Society for Optical Engineering), Sensor Fusion and Decentralized Control in Robotic Systems III*, pages 348–359, Oct. 2000.

G. Hoffmann, S. Waslander, and C. Tomlin. Distributed cooperative search using information-theoretic costs for particle filters, with quadrotor applications. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Aug. 2006.

B. J. P. Horn. Tsai's camera calibration method revisited.
Online: http://people.csail.mit.edu/bkph/articles/Tsai_Revisited.pdf, 2000.

S. J. Julier. The scaled unscented transformation. In *American Control Conference*, volume 6, pages 4555–4559, Anchorage, AK, USA, May 2002.

T. Kailath, A. H. Sayed, and B. Hassibi. *Linear Estimation*. Prentice Hall, 2000.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the American Society of Mechanical Engineering — Journal Basic Engieering*, 82 (Series D):35–45, Mar. 1960.

G. Mathews and H. Durrant-Whyte. Decentralised optimal control for reconnaissance. In *In Proceedings of the Conference on Information, Decision and Control*, Adelaide, Australia, Feb. 2007.

A. Ryan, X. Xiao, S. Rathinam, J. Tisdale, D. Caveney, R. Sengupta, and J. K. Hedrick. A modular software infrastructure for distributed control of collaborating unmanned aerial vehicles. In *Proceedings of the AIAA Guidance, Navigation, and Control Conference and Exhibit*, Aug. 2006.

J. Tisdale, A. Ryan, M. Zennaro, X. Xiao, D. Caveney, S. Rathinam, J. K. Hedrick, and R. Sengupta. The software architecture of the berkeley UAV platform. In *Proceedings of the Conference on Control Applications*, 2006.

R. Vidal, O. Shakernia, H. J. Kim, H. Shim, and S. Sastry. Multi-agent probabilistic pursuit-evasion games with unmanned ground and aerial vehicles. *IEEE Journal of Robotics and Automation*, 18:662–669, 2002.

E. Wong, F. Bourgault, and T. Furukawa. Multi-vehicle bayesian search for multiple lost targets. In *Proceedings of the IEEE international conference on robotics and automation*, 2005.

Notation

Abbreviations and Acronyms

Abbreviation	Meaning
BLUE	Best Linear Unbiased Estimator
DARPA	Defense Advanced Research Projects Agency
EKF	Extended Kalman Filter
FA	False Alarm
GLR	Generalized Likelihood Ratio
GLRT	Generalized Likelihood Ratio Test
HMM	Hidden Markov Model
IMU	Inertial Measurement Model
KF	Kalman Filter
LIDAR	LIght Detection And Ranging
LR	Likelihood Ratio
LRT	Likelihood Ratio Test
LS	Least Squares
M	Miss
MAP	Maximum APosteriori
MC	Monte Carlo
ML	Maximum Likelihood
MPF	Marginalized Particle Filter
NP	Neyman-Pearson
PDF	Probability Density Function
PF	Particle Filter
RBPF	Rao-Blackwellized Particle Filter
SLAM	Simultaneous Localization And Mapping
UAV	Unmanned Aerial Vehicle
UMP	Uniformly Most Powerful

Symbols and Mathematical Notation

Notation	Meaning
$\mathbb{Y}$	Stacked variables in a window of length L, $\mathbb{Y} = \left(y_{t-L+1}^T, \dots, y_t^T\right)^T$.
$y_{1:t}$	The set of variables $\{y_1, y_2, \dots, y_t\}$.
$\triangleq$	Defined as.
$\odot$	Quaternion product, see (B.11).
$x \sim y$	x is distributed as y.
$\arg\max_x f(x)$	The x maximizing $f(x)$.
$\arg\min_x f(x)$	The x minimizing $f(x)$.
$\mathsf{Cov}(x)$	Covariance of x.
$\delta(\,\cdot\,)$	Dirac delta.
$\mathrm{diag}(x_1, \dots, x_n)$	A diagonal matrix with x_i in the diagonal.
e_t	Measurement noise at time t.
$\mathsf{E}(x)$	Expected value of x.
$f(\,\cdot\,)$	State propagation function.
f_t	Deterministic but unknown fault at time t.
$\bar{H}^s$	System matrix for inputs in batch formulation, see (2.9).
$\mathcal{H}_i$	A hypothesis.
$h(\,\cdot\,)$	Measurement function.
$L(\,\cdot\,)$	Likelihood ratio.
L	Window size.
$\mathsf{N}(x; \mu, \Sigma)$	Gaussian PDF for mean μ and covariance Σ.
$\mathsf{N}(\mu, \Sigma)$	Gaussian distribution with mean μ and covariance Σ.
n_x	Dimension of the variable x.
$\mathcal{O}$	Extended observability matrix, see (2.7).
$\mathcal{O}(\,\cdot\,)$	Denotes computational complexity.
$p(\,\cdot\,)$	Probability density function.
$\hat{p}(\,\cdot\,)$	Approximative probability density function.
$\pi(\,\cdot\,)$	Proposal distribution.
Q_t	Covariance of process noise at time t.
R_t	Covariance of measurement noise at time t.
$\mathrm{rank}(A)$	Rank of A.
A^T	A transposed.
u_t	Known input at time t.
$\mathsf{Var}(x)$	Variance of x.
v_t	Process noise at time t.
x_t	State at time t.
$\hat{x}_{t\|\tau}$	Estimate of x_t given the measurements $y_{1:\tau}$.
χ_ν^2	Chi-square distribution with ν degrees of freedom.
$\chi_\nu'^2(\lambda)$	Noncentral chi-square distribution with ν degrees of freedom and noncentrality parameter λ.
y_t	Measurement at time t.

PhD Dissertations
Division of Automatic Control
Linköping University

M. Millnert: Identification and control of systems subject to abrupt changes. Thesis No. 82, 1982. ISBN 91-7372-542-0.

A. J. M. van Overbeek: On-line structure selection for the identification of multivariable systems. Thesis No. 86, 1982. ISBN 91-7372-586-2.

B. Bengtsson: On some control problems for queues. Thesis No. 87, 1982. ISBN 91-7372-593-5.

S. Ljung: Fast algorithms for integral equations and least squares identification problems. Thesis No. 93, 1983. ISBN 91-7372-641-9.

H. Jonson: A Newton method for solving non-linear optimal control problems with general constraints. Thesis No. 104, 1983. ISBN 91-7372-718-0.

E. Trulsson: Adaptive control based on explicit criterion minimization. Thesis No. 106, 1983. ISBN 91-7372-728-8.

K. Nordström: Uncertainty, robustness and sensitivity reduction in the design of single input control systems. Thesis No. 162, 1987. ISBN 91-7870-170-8.

B. Wahlberg: On the identification and approximation of linear systems. Thesis No. 163, 1987. ISBN 91-7870-175-9.

S. Gunnarsson: Frequency domain aspects of modeling and control in adaptive systems. Thesis No. 194, 1988. ISBN 91-7870-380-8.

A. Isaksson: On system identification in one and two dimensions with signal processing applications. Thesis No. 196, 1988. ISBN 91-7870-383-2.

M. Viberg: Subspace fitting concepts in sensor array processing. Thesis No. 217, 1989. ISBN 91-7870-529-0.

K. Forsman: Constructive commutative algebra in nonlinear control theory. Thesis No. 261, 1991. ISBN 91-7870-827-3.

F. Gustafsson: Estimation of discrete parameters in linear systems. Thesis No. 271, 1992. ISBN 91-7870-876-1.

P. Nagy: Tools for knowledge-based signal processing with applications to system identification. Thesis No. 280, 1992. ISBN 91-7870-962-8.

T. Svensson: Mathematical tools and software for analysis and design of nonlinear control systems. Thesis No. 285, 1992. ISBN 91-7870-989-X.

S. Andersson: On dimension reduction in sensor array signal processing. Thesis No. 290, 1992. ISBN 91-7871-015-4.

H. Hjalmarsson: Aspects on incomplete modeling in system identification. Thesis No. 298, 1993. ISBN 91-7871-070-7.

I. Klein: Automatic synthesis of sequential control schemes. Thesis No. 305, 1993. ISBN 91-7871-090-1.

J.-E. Strömberg: A mode switching modelling philosophy. Thesis No. 353, 1994. ISBN 91-7871-430-3.

K. Wang Chen: Transformation and symbolic calculations in filtering and control. Thesis No. 361, 1994. ISBN 91-7871-467-2.

T. McKelvey: Identification of state-space models from time and frequency data. Thesis No. 380, 1995. ISBN 91-7871-531-8.

J. Sjöberg: Non-linear system identification with neural networks. Thesis No. 381, 1995. ISBN 91-7871-534-2.

R. Germundsson: Symbolic systems – theory, computation and applications. Thesis No. 389, 1995. ISBN 91-7871-578-4.

P. Pucar: Modeling and segmentation using multiple models. Thesis No. 405, 1995. ISBN 91-7871-627-6.

H. Fortell: Algebraic approaches to normal forms and zero dynamics. Thesis No. 407, 1995. ISBN 91-7871-629-2.

A. Helmersson: Methods for robust gain scheduling. Thesis No. 406, 1995. ISBN 91-7871-628-4.

P. Lindskog: Methods, algorithms and tools for system identification based on prior knowledge. Thesis No. 436, 1996. ISBN 91-7871-424-8.

J. Gunnarsson: Symbolic methods and tools for discrete event dynamic systems. Thesis No. 477, 1997. ISBN 91-7871-917-8.

M. Jirstrand: Constructive methods for inequality constraints in control. Thesis No. 527, 1998. ISBN 91-7219-187-2.

U. Forssell: Closed-loop identification: Methods, theory, and applications. Thesis No. 566, 1999. ISBN 91-7219-432-4.

A. Stenman: Model on demand: Algorithms, analysis and applications. Thesis No. 571, 1999. ISBN 91-7219-450-2.

N. Bergman: Recursive Bayesian estimation: Navigation and tracking applications. Thesis No. 579, 1999. ISBN 91-7219-473-1.

K. Edström: Switched bond graphs: Simulation and analysis. Thesis No. 586, 1999. ISBN 91-7219-493-6.

M. Larsson: Behavioral and structural model based approaches to discrete diagnosis. Thesis No. 608, 1999. ISBN 91-7219-615-5.

F. Gunnarsson: Power control in cellular radio systems: Analysis, design and estimation. Thesis No. 623, 2000. ISBN 91-7219-689-0.

V. Einarsson: Model checking methods for mode switching systems. Thesis No. 652, 2000. ISBN 91-7219-836-2.

M. Norrlöf: Iterative learning control: Analysis, design, and experiments. Thesis No. 653, 2000. ISBN 91-7219-837-0.

F. Tjärnström: Variance expressions and model reduction in system identification. Thesis No. 730, 2002. ISBN 91-7373-253-2.

J. Löfberg: Minimax approaches to robust model predictive control. Thesis No. 812, 2003. ISBN 91-7373-622-8.

J. Roll: Local and piecewise affine approaches to system identification. Thesis No. 802, 2003. ISBN 91-7373-608-2.

J. Elbornsson: Analysis, estimation and compensation of mismatch effects in A/D converters. Thesis No. 811, 2003. ISBN 91-7373-621-X.

O. Härkegård: Backstepping and control allocation with applications to flight control. Thesis No. 820, 2003. ISBN 91-7373-647-3.

R. Wallin: Optimization algorithms for system analysis and identification. Thesis No. 919, 2004. ISBN 91-85297-19-4.

D. Lindgren: Projection methods for classification and identification. Thesis No. 915, 2005. ISBN 91-85297-06-2.

R. Karlsson: Particle Filtering for Positioning and Tracking Applications. Thesis No. 924, 2005. ISBN 91-85297-34-8.

J. Jansson: Collision Avoidance Theory with Applications to Automotive Collision Mitigation. Thesis No. 950, 2005. ISBN 91-85299-45-6.

E. Geijer Lundin: Uplink Load in CDMA Cellular Radio Systems. Thesis No. 977, 2005. ISBN 91-85457-49-3.

M. Enqvist: Linear Models of Nonlinear Systems. Thesis No. 985, 2005. ISBN 91-85457-64-7.

T. B. Schön: Estimation of Nonlinear Dynamic Systems — Theory and Applications. Thesis No. 998, 2006. ISBN 91-85497-03-7.

I. Lind: Regressor and Structure Selection — Uses of ANOVA in System Identification. Thesis No. 1012, 2006. ISBN 91-85523-98-4.

J. Gillberg: Frequency Domain Identification of Continuous-Time Systems Reconstruction and Robustness. Thesis No. 1031, 2006. ISBN 91-85523-34-8.

M. Gerdin: Identification and Estimation for Models Described by Differential-Algebraic Equations. Thesis No. 1046, 2006. ISBN 91-85643-87-4.

C. Grönwall: Ground Object Recognition using Laser Radar Data – Geometric Fitting, Performance Analysis, and Applications. Thesis No. 1055, 2006. ISBN 91-85643-53-X.

A. Eidehall: Tracking and threat assessment for automotive collision avoidance. Thesis No. 1066, 2007. ISBN 91-85643-10-6.

F. Eng: Non-Uniform Sampling in Statistical Signal Processing. Thesis No. 1082, 2007. ISBN 978-91-85715-49-7.

E. Wernholt: Multivariable Frequency-Domain Identification of Industrial Robots. Thesis No. 1138, 2007. ISBN 978-91-85895-72-4.

D. Axehill: Integer Quadratic Programming for Control and Communication. Thesis No. 1158, 2008. ISBN 978-91-85523-03-0.

G. Hendeby: Performance and Implementation Aspects of Nonlinear Filtering. Thesis No. 1161, 2008. ISBN 978-91-7393-979-9.

J. Sjöberg: Optimal Control and Model Reduction of Nonlinear DAE Models. Thesis No. 1166, 2008. ISBN 978-91-7393-964-5.